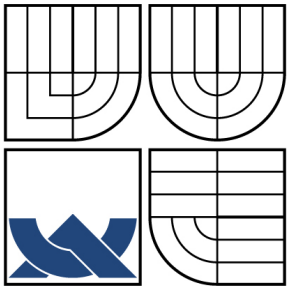
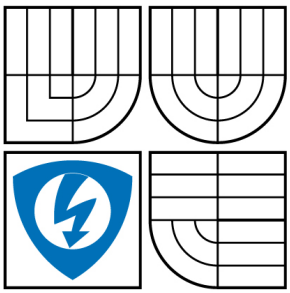


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

URČENÍ PARAMETRŮ POHYBU ZE SNÍMKŮ KAMERY

DETERMINATION OF MOTION PARAMETERS IN MACHINE VISION

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

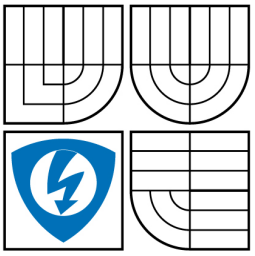
AUTOR PRÁCE — Bc. STANISLAV DUŠEK

AUTHOR

VEDOUCÍ PRÁCE — Ing. ILONA KALOVÁ, Ph.D.

SUPERVISOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student:	Bc. Stanislav Dušek	***ID:***	83827
Ročník:	2	***Akademický rok:***	2008/2009

NÁZEV TÉMATU:

Určení parametrů pohybu ze snímků kamery

POKYNY PRO VYPRACOVÁNÍ:

Úkolem studenta je nastudovat problematiku detekce pohybu ze snímků kamery a navrhnout algoritmus, který určí změny polohy kamery v X-ové a Y-ové ose na základě vzájemné analýzy dvou snímků při statické snímané scéně. Předpokládá se praktická realizace.

DOPORUČENÁ LITERATURA:

Hlaváč, V., Šonka, M.: Počítačové vidění. Praha: Grada, 1992.
Haußecker H., Geißler P.: Handbook of Computer Vision and Applications. San Diego: Academic press, 1999.

Termín zadání:	9.2.2009	***Termín odevzdání:***	25.5.2009

Vedoucí práce: Ing. Ilona Kalová, Ph.D.

prof. Ing. Pavel Jura, CSc.
Předseda oborové rady

LICENČNÍ SMLOUVA

POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení: Stanislav Dušek
Bytem: Křepiny15(obec Řečice), 39601 Humpolec
Narozen/a (datum a místo): 22.05.1985 Pelhřimov
(dále jen „autor“)

a

2. Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií
se sídlem Údolní 244/53, 602 00, Brno
jejímž jménem jedná na základě písemného pověření děkanem fakulty:
prof. Ing. Pavel Jura, CSc.
(dále jen „nabyvatel“)

Čl. 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
   - □ disertační práce
   - ■ diplomová práce
   - □ bakalářská práce
   - □ jiná práce, jejíž druh je specifikován jako

     .....

   (dále jen VŠKP nebo dílo)

Název VŠKP:	Určení parametrů pohybu ze snímků kamery
Vedoucí/ školitel VŠKP:	Ing. Ilona Kalová, Ph.D.
Ústav:	Ústav automatizace a měřicí techniky
Datum obhajoby VŠKP:	11.06.2009

VŠKP odevzdal autor nabyvateli v*:

- ■ tištěné formě – počet exemplářů ……1……..
- ■ elektronické formě – počet exemplářů ……1……..

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2

Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti
   - ■ ihned po uzavření této smlouvy
   - □ 1 rok po uzavření této smlouvy
   - □ 3 roky po uzavření této smlouvy
   - □ 5 let po uzavření této smlouvy
   - □ 10 let po uzavření této smlouvy

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FAKULTA ELEKTRONIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

Určení parametrů pohybu ze snímků kamery

Diplomová práce

Specializace studia: Kybernetika, automatizace a měření
Student: Bc. Stanislav Dušek
Vedoucí práce: Ing. Ilona Kalová, Ph.D.

Abstrakt :

Tato práce pojednává o parametrizování pohybu kamery v rovině. První část práce je věnována úvodu do problematiky trasování pohybu kamery, zejména se věnuje možnostem, jak nalézt vektor posunu mezi dvěma po sobě jdoucími obrázky. V práci je popsán algoritmus GoodFeatruresToTrack, který vyhledává význačné body v obraze. Cílem je najít v obraze vhodné body k trasování, zmenšit objem zpracovávaných dat a připravit data pro algoritmus optického toku, konkrétně metodu Lukas-Kanade.

Dále se práce zabývá zpracováním získaných odhadů optických toků, od filtrace dat pomocí mediánu, až po výpočet transformací mezi snímky pomocí matice homogenní transformace, která zachycuje změny mezi snímky v afinním prostoru. Výsledkem zpracování jsou souřadnice popisující posun mezi snímky v osách X a Y a trajektorie pohybu.

Správná funkčnost navrženého algoritmu byla ověřena pomocí programu CameraTracer, který je naprogramován v C++ a využívá knihovnu OpenCV. Program byl testován při snímání různých scén a pohybů kamery.

Klíčová slova:

Odhad pohybu kamery, optický tok, Lucas-Kanade, trasování pohybu, určení parametrů pohybu, význačné body, Good features to track, homogenní transformace, dírková kamera, zpracovávání obrazu, počítačové vidění, Open CV.

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL, MEASUREMENT AND INSTRUMENTATION

Determination of Motion Parameters in Machine Vision

Master Thesis

Specialisation of study: Cybernetics, control and Measurements
Student: Bc. Stanislav Dušek
Supervisor: Ing. Ilona Kalová, Ph.D.

Abstract:

This thesis discusses determination of the camera motion parameters in the plane. At first there are introduced the basics of motion tracking. It is focused on to find out displacement between two input images. The algorithm GoodFeatruresToTrack, which traces the most significant points in a first image, is described. The aim is to find suitable points, which will be easily tracked in the next image, to reduce the data volume and prepare the input information (array of significant points) for the algorithm of the optical flow, concretely Lucas-Kanade algorithm.

In the second part is dealt with processing and utilization of estimation of the optical flow. There is median filtration and further is described computation of homogenous transformation, which describes all affine transformations in the affine space. The result is represented as the coordinates, which describe the shift between the two input images as X-axis and Y-axis values.

In the project is used the library Open Computer Vision.

Keywords:

Estimation of camera motion, optical flow, Lucas-Kanade, tracking of motion, determination of motion parameters, significant point, Good features to track, homogenous transformation, pinhole camera, image processing, computer vision, Open CV.

DUŠEK, S. *Určení parametrů pohybu ze snímků kamery*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 76 s. Vedoucí diplomové práce Ing. Ilona Kalová, Ph.D.

Prohlášení

„Prohlašuji, že svou diplomovou práci na téma "Určení parametrů pohybu ze snímků kamery" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.“

V Brně dne: **25. května 2009** ……

podpis autora

Poděkování

Děkuji vedoucímu diplomové práce Ing. Iloně Kalové, Ph.D za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne: **25. května 2009** ……

podpis autora

OBSAH:

SEZNAM OBRÁZKŮ :

1. ÚVOD

1.1 PODROBNĚJŠÍ POPIS PROBLÉMU

Úkolem diplomové práce je nalézt a popsat algoritmus, který je schopen ze dvou po sobě jdoucích snímků z videosekvence zjistit jejich vzájemný posun. Předpokládá se, že ve scéně nejsou žádné pohyblivé objekty, jediné co se pohybuje je kamera, a ta se pohybuje rovnoběžně se snímaným povrchem. Vzhledem k plánovanému využití pro sledování pohybu vrtulníku či mobilního robota se předpokládá, že snímaná scéna bude relativně chudá na rozmanité předměty, spíše budou zaznamenané různé povrchy, jako zem, podlaha, silnice. Navržený algoritmus by měl být schopen rozpoznat, jakým směrem se kamera pohybuje.

1.2 ODHAD POHYBU KAMERY

Je postup, který se snaží co nejlépe popsat vektor pohybu popisující změnu jednoho obrazu na druhý, nejčastěji ve videosekvenci. Problémem bývá, že skutečný pohyb se uskutečňuje ve 3D světě, ale obraz zachycený kamerou je pouze 2D. Pohybový vektor se může vztahovat k celé scéně nebo jen k určitým objektům nebo může být i vyjádřený pro každý pixel samostatně. Pohybový vektor může být v nejjednodušším případě reprezentován například translačním modelem, další modely vychází z pohybu skutečné videokamery tj. jako rotace a translace ve 3D a „zoom“.

V naznačeném případě se počítá s konstantní vzdáleností snímaného povrchu od kamery. Tím se úloha zjednodušuje na hledání pohybu v rovině, požadováno je pouze hledání pohybu v osách X a Y. Pohyb se bude odhadovat jako kombinace translace a rotace. Pohyb objektu bude snímán kamerou se snímkovací frekvencí 15 nebo 30 snímků za sekundu. Lze předpokládat, že jeden bod se mezi dvěma obrazy posune jen o určitou relativně malou vzdálenost.

1.3 METODY NALEZENÍ POHYBOVÉHO VEKTORU

Možnosti jak nalézt vektor pohybu jsou v zásadě dvě, první možností jsou metody tzv. přímé, ty jsou pixelově orientované, patří sem například blokové porovnávání, fázová korelace, další matematické metody typu SAD (Sum of Absolute Differences). Druhou možností jsou metody vycházející z výrazných bodů v obraze, například použití Harrisonova detektoru a následného hledání korespondencí.

Další velmi rozsáhlou kategorií jsou metody optického toku, ty lze díky rozmanitosti a obsáhlosti považovat za vlastní kategorii. Nejznámější jsou metody Lucas–Kanade a Horn-Schunck.

1.4 POSTUP SLEDOVÁNÍ POHYBU KAMERY

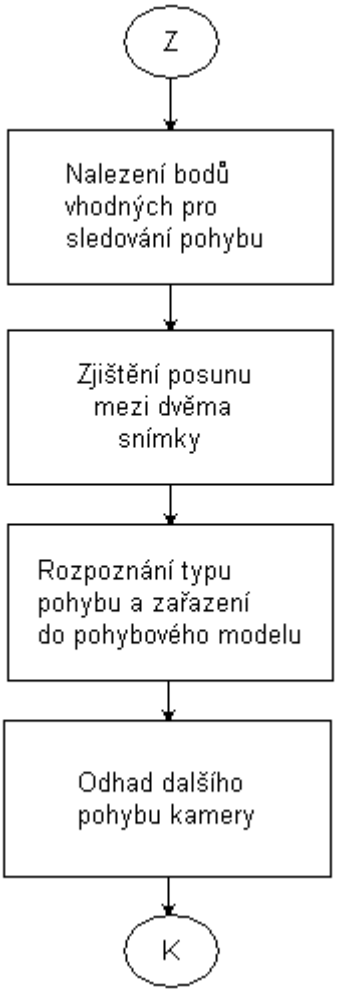
Aby bylo možné odhalit pohyb obrazu, musí se ve snímku nacházet rozlišitelné objekty, případně jejich části nebo alespoň struktury s malou hustotou opakování. Určitě podle některých částí snímků (bodů či objektů v nich obsažených) je jednodušší pohyb poznat. Intuitivně to jsou ty, které se od svého okolí nejvíce odlišují, tvoří rohy, výstupky, „šmouhy“. Hledáním těchto bodů se zabývají rohové detektory a nalezené body se nazývají významné body v obraze (viz kapitola 2). Zaměřením na vybranou skupinu nejvíce významných bodů lze dosáhnout větší efektivity navazujících algoritmů .

Dalším krokem je samotné zjištění posunu mezi obrazy, s ohledem na snímanou scénu, ve které převládají podobně členěné povrchy, není vhodné používat přímé metody. Bylo by velmi náročné zvolit vhodný parametr podobnosti a vedlo by to k velkému množství nepřesností. Jinou možností je použít rohový detektor a porovnávat pozice význačných bodů zjištěných v analyzovaných obrazech, bohužel pro složitě strukturované obrazy by to vedlo ke složitému řešení korespondenčního problému. Pro jeden bod v prvním obraze by připadalo v úvahu velké množství možných bodů v druhém obraze. Jako vhodnější vychází použití optického toku (viz kapitola 3), ten si poradí i s poměrně složitými strukturami snímku. Omezení, v podobě maximální velikosti elementárního pohybu, není úplně na závadu, zabraňuje totiž zbytečným a zdlouhavým výpočtům, které by většinou nevedly ke

správnému odhadu pohybu. Jako výsledek tohoto kroku se předpokládá vektor, ve kterém je každému bodu, pro který se povedlo zjistit pohyb, přirazena jeho pozice a směr posunu.

Poté se provede statistické vyhodnocení zjištěných údajů a zvolí se vhodný popis pohybu. Například když jsou všechny zjištěné posuny v jednom směru a přibližně stejné velikosti, lze tedy použít translační popis. Pokud ne předpokládá se, že převažuje rotační pohyb. Nalezený pohyb se přidá do vytvářené trajektorie pohybu.

Na závěr se algoritmus pokusí odhadnout další přepokládanou trajektorii pohybu kamery.



Obrázek 1: Přehledové schéma zpracování odhadu pohybu kamery

2. VYHLEDÁNÍ VÝRAZNÝCH BODŮ V OBRAZE

2.1 HARRISONŮV – STEPHENSŮV DETEKTOR ROHŮ

Algoritmus je založen na lokální autokorelační funkci I(*x*,*y*) jsou hodnoty pixelů obrazu a *Δx, Δy* jsou elementární posuvy z množiny **D**, představuje okénkovou funkci [8]. Pomocí Taylorova rozvoje aproximujeme
I(x + Δx,y + Δy) jako:

$$I(x+\Delta x, y+\Delta y) \approx I(x,y) + \left(\frac{\partial I(x,y)}{\partial x}, \frac{\partial I(x,y)}{\partial y}\right)^T \begin{pmatrix}\Delta x \\ \Delta y\end{pmatrix} \tag{2.1}$$

Dosazením funkce:

$$F_{\min}(x,y) \approx \min_{(\Delta x,\Delta y)\in D} \sum_{(x,y)\in O} (\Delta x,\Delta y)\left(\frac{\partial I(x,y)}{\partial x}, \frac{\partial I(x,y)}{\partial y}\right)\left(\frac{\partial I(x,y)}{\partial x}, \frac{\partial I(x,y)}{\partial y}\right)^T$$

$$= \min_{(\Delta x,\Delta y)\in D} (\Delta x,\Delta y) M \begin{pmatrix}\Delta x \\ \Delta y\end{pmatrix} \tag{2.2}$$

kde matice ***M*** je označována jako *autokorelační*

$$M = \sum_{(x,y)\in D}\left(\frac{\partial I(x,y)}{\partial x}, \frac{\partial I(x,y)}{\partial y}\right)\begin{pmatrix}\frac{\partial I(x,y)}{\partial x} \\ \frac{\partial I(x,y)}{\partial y}\end{pmatrix} = \sum_{(x,y)\in D}\begin{bmatrix} I_x^2 & I_x \cdot I_y \\ I_x \cdot I_y & I_x^2 \end{bmatrix} \tag{2.3}$$

Kde I_x a I_y jsou parciální derivace pro dané okolí **D**. Matice **M**, lépe řečeno její vlastní čísla odráží hladkost okolí zkoumaného bodu a podle parametrů vlastních čísel (označíme je jako λ_1 a λ_2) lze poznat, jestli je v daném bodě roh a jak je významný.

Mohou nastat 3 případy porovnání hodnot λ_1 a λ_2:

- Obě jsou relativně malá, potom je okolí bodu relativně ploché a tzn. téměř konstantní hodnoty jasu v okolí

- Jedno vlastní číslo je malé a druhé velké, potom se jedná o hranu v jednom směru
- Obě vlastní čísla jsou velká, potom je v okolí bodu významná změna jasu ve více směrech, jedná se tedy o významný bod - roh, nebo o izolovaný bod

Dále se počítá indikátor přítomnosti rohu R:

$$R = \det M - \kappa \cdot eigM = \det\begin{bmatrix} I_x^2 & I_x \cdot I_y \\ I_x \cdot I_y & I_x^2 \end{bmatrix} - \kappa \cdot eig\begin{bmatrix} I_x^2 - \lambda & I_x \cdot I_y \\ I_x \cdot I_y & I_x^2 - \lambda \end{bmatrix} \quad (2.4)$$

$$R = \lambda_1\lambda_2 - \kappa(\lambda_1 + \lambda_2)^2 \quad (2.5)$$

Přesné počítání vlastních čísel je poměrně náročná operace (obsahuje matematické operace jako druhá odmocnina), proto se místo toho použije výpočet funkce R_c :

$$R_c = \lambda_1\lambda_2 - \kappa(\lambda_1 + \lambda_2)^2 = \det(M) - \kappa \cdot trace^2(M) \quad (2.6)$$

Kde funkce *trace* znamená součet členů ležících na hlavní diagonále matice ***M*** a hodnota κ je empiricky zjištěná hodnota, v literatuře se většinou uvádí hodnota 0,040-0,150 . Hodnota 0,04 je používána nejčastěji.

2.2 ALGORITMUS - GOOD FEATURES TO TRACK

2.2.1 Princip funkce algoritmu

Hledá nejvýraznější body v obraze, vhodné k trasování pohybu. Algoritmus [3] vymysleli Jianbo Shi and Carlo Tomasi a poprvé prezentovali na konferenci počítačového vidění v roce 1994. Hlavním úkolem, který si tento algoritmus klade za cíl, je nalézt dostatečně velkou množinu dostatečně kvalitních bodů, vhodných pro sledování pohybu mezi dvěma obrázky. Algoritmus musí být schopný najít kvalitní body v různých vstupních obrázcích bez předchozí znalosti jejich vlastností. Střední hodnota počtu nalezených bodů by se měla pohybovat v rozumných mezích, aby nedocházelo k velkým nárůstům výpočetní náročnosti odhadu pohybu mezi snímky.

Algoritmus vyhledávání vhodných bodů vychází z Harrisonova-Stephensova detektoru rohů (viz 2.1). Ovšem s ohledem na možnou změnu obrazu při afinních transformacích upravuje problematický výpočet rovnice (3.6) na:

$$R=\min(\lambda 1,\lambda 2) \qquad (2.7)$$

Jako hodnota parametru *R* v definovaném okolí se uchovává menší hodnota z vlastních čísel. Protože pokud menší hodnota z vlastních čísel vykazuje dostatečně velkou odezvu - větší než práh, potom i hrana v tomto místě bývá významná. Takto se přiřadí hodnota R každému pixelu v obraze. Přitom se zaznamenává největší hodnota parametru R, tj. nejvýraznější roh v celém obraze. Následně se podle zadaného koeficientu kvality vyberou body, které jsou kvalitativně lepší než mez, zadaná poměrem k nejlepšímu bodu např. 0,15 (tj. 15%). Dále je počet ukládaných bodů omezen hodnotou maximálního počtu N_{max}. To znamená, že se vyberou jen dostatečně kvalitní výrazné body, kterých může být i relativně málo, záleží na zvoleném kvalitativním měřítku nejvíce však N_{max}. Na závěr se odstraní body, které jsou blíže než zadaná minimální vzdálenost, z nich se zanechá jen ten nejsilnější. Tímto způsobem je funkce algoritmu realizována, jako knihovní funkce OpenCV (viz kapitola 6), kterou jsem ve svém programu použil. Výsledek algoritmu je znázorněn na obrázku Obrázek 15, oblast, ve které jsou hledány body, je ohraničena modrým obdélníkem.

3. OPTICKÝ TOK

3.1 ZÁKLADNÍ METODY POČÍTÁNÍ OPTICKÉHO TOKU

Existují dva základní přístupy, jak počítat optický tok, první metoda se nazývá ***„dense optical flow"*** – optický tok se počítá pro všechny body v obraze. Vezmeme-li v úvahu, že se v obraze vyskytují i celistvé jednolité plochy – například bílý papír, je velmi obtížné počítat optický tok pro všechny body. V praxi se to potom řeší tak, že obtížně počitatelné body se interpolují pomocí bodů, jejichž optický tok lze počítat snáze. Ani tak nemusí být řešení jednoznačné, výsledný algoritmus je poměrně složitý a hlavně časově náročný. Představitelem tohoto typu algoritmu je metoda Horn-Schunck.

Druhou možností jsou algoritmy typu ***„sparse optical flow"***, tyto algoritmy vychází z předem definované množiny bodů, které by měly být relativně jednoduše sledovatelné. Způsoby jak vybrat vhodné body jsou popsány v kapitole 2. Pro praktické využití je velmi významné to, že časová náročnost je ve většině případů podstatně menší než u algoritmů typu *„dense optical flow"*. Nejznámějším algoritmem tohoto typu je Lucas-Kanade optical flow, podrobně je tento algoritmus rozebrán v další kapitole.

3.2 TRASOVACÍ ALGORITMY VYCHÁZEJÍCÍ Z OPT. TOKU

Důležité vlastnosti každého trasovacího algoritmu jsou přesnost a robustnost. Přesnost souvisí s volbou integračního okna při výpočtu optického toku. Malé integrační okno je vhodné pro „nerozmazané" obrazy s velkým množstvím detailů, zejména pokud se ve scéně pohybuje více objektů rozdílnou a relativně vysokou rychlostí.

Robustnost závisí na míře citlivosti vůči změnám osvětlení a velikosti pohybující se části obrazu. Pro zvládání velkých pohybů je vhodné využít větší integrační okno. Kterou z těchto dvou složek upřednostnit, to záleží na konktrétní aplikaci. Jinou možností, jak řešit tento rozpor, je využití pyramidové implementace.

Iterativní algoritmus optického toku Lucas-Kanade s pyramidovou implementací [4], řeší výpočet trasování s dostatečnou přesností.

3.3 FORMULACE PROBLÉMU:

Máme dva digitální (nakvantované) šedotónové obrázky **I** a **J**. Souřadnice obrazových bodů lze vyjádřit pomocí *x,y*, tedy obrazy lze popsat pomocí *I(x,y) a J(x,y).* A lze s nimi pracovat jako s maticemi diskrétních hodnot. Obraz **I** je první obraz, **J** je druhý obraz a oba mají stejnou velikost. Úkolem trasování je bod z obrázku **I** nalézt v obrázku **J**, respektive nalézt jeho souřadnice. Jestliže vektor $u = [u_x \quad u_x]^T$ odpovídá bodům v prvním obrázku, potom je cílem nalézt vektor $u = [u_x + d_x \quad u_y + d_y]^T$ ve druhém obrázku. Vektor $d = [d_x \quad d_y]^T$ je obrazová rychlost, spíše známá jako optický tok. Protože se zde vyskytuje „aperaturní problém“, je nezbytné zabývat se pojmem podobnosti v 2D sousedství. To se omezí hodnotami ω_x a ω_y a potom okolí bodu velikosti $(2\omega_x + 1) \times (2\omega_y + 1)$ se označuje jako integrační okno. Takto vznikne optický tok omezený reziduální funkcí:

$$\varepsilon(d) = \varepsilon(d_x, d_y) = \sum_{x=u_x-\omega_x}^{u_x+\omega_x} \sum_{y=u_y-\omega_y}^{u_y+\omega_y} (I(x,y) - J(x+d_x, y+d_y))^2 \quad (3.1)$$

4. ALGORITMUS OPTICKÉHO TOKU LUCAS-KANADE

Popisovaný algoritmus se plným názvem jmenuje: algoritmus optického toku s rozptýlenými body využívající iterační pyramidový algoritmus Lucas-Kanade. Dále je používán pouze název Lucas-Kanade nebo zkratka L-K.

4.1 OPTICKÝ TOK LUCAS-KANADE

Byl představen v roce 1981 [4]. Při výpočtu využívá pouze malé okolí náležející výrazným bodům. Problémem tohoto přístupu je, že lze počítat pouze malou změnu optického toku a to pouze do velikosti integračního okna. Tato nevýhoda byla ovšem vyřešena použitím rozšíření v podobě pyramidové implementace, ta umožňuje zachytit i daleko větší pohyby než je samotné integrační okno (vysvětleno v části Pyramidová implementace trasovacího algoritmu).

4.1.1 Základní podmínky, z nichž algoritmus L-K vychází

- Konstantní jas – pixely ve scéně se skokově nemění, mezi jednotlivými snímky si uchovávají relativně stejný jas [9].
- Malé pohyby – rychlost změny pohybu sledovaných bodů je relativně malá, to znamená, že dochází pouze k přírůstkům pohybu mezi jednotlivými snímky. [9]
- Prostorová koherence – většina okolních pixelů leží v té samé rovině a vykonávají ten samý pohyb. [9]

4.1.2 Blíže k jednotlivým podmínkám

První požadavek - konstantní jas říká, že během sledování se nemění intenzita pixelu. Matematicky to lze zapsat jako [9] :

$$f(x,t) \equiv I(x(t),t) = I(x(t+dt),t+dt) \qquad (3.2)$$

Nebo:

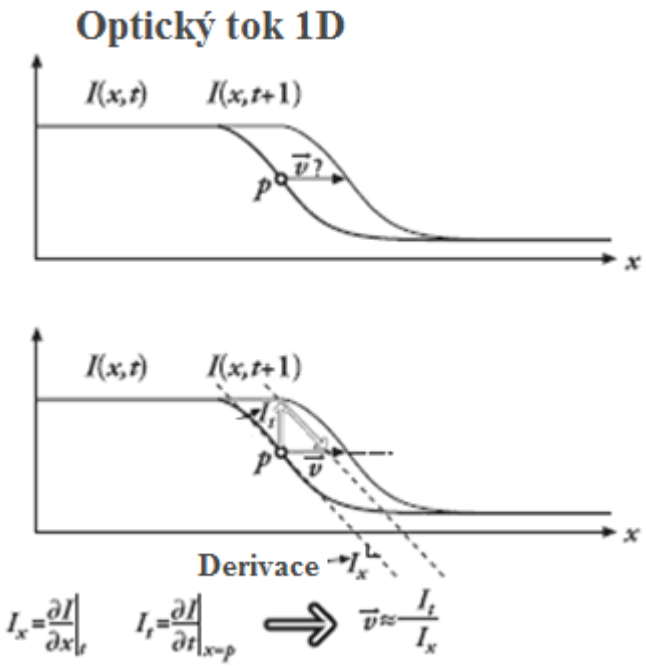
$$\frac{\delta f(x)}{\delta t} = 0 \qquad (3.3)$$

Druhá podmínka požaduje, aby pohyb mezi jednotlivými snímky byl diferenciálně malý. Předpokládejme, že první podmínka je splněna - jas obrazu se nemění. Vezměme si funkci jasu f(x,t), předpokládá se, že průběh x je závislý na t, **I**(x(t),t). Potom musí platit, že:

$$\underbrace{\frac{\delta I}{\delta t}\bigg|_t}_{I_x}\underbrace{\left(\frac{\delta x}{\delta t}\right)}_{v}+\underbrace{\frac{\delta I}{\delta t}\bigg|_{x(t)}}_{It}=0 \qquad (3.4)$$

Kde I_x je derivace prvního obrazu, I_t je derivace mezi obrazy za čas t a v je změna pohybu, kterou hledáme. Pro tuto jednoduchou 1D funkci platí [9] :

$$v=-\frac{I_t}{I_x} \qquad (3.5)$$



Obrázek 2: Znázornění výpočtu optického toku pro 1D funkci

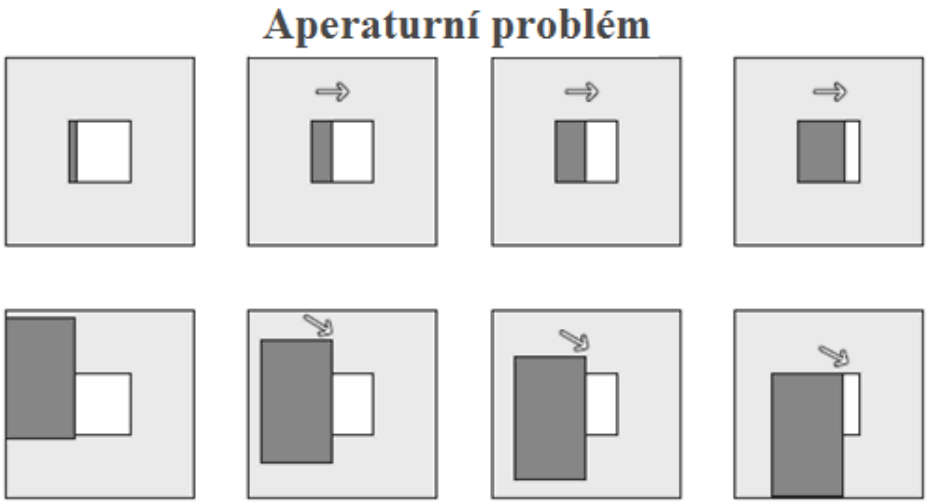
Nyní je potřeba pro zadanou 1D funkci nalézt změnu pohybu v, ta se většinou označuje jako optický tok. Uvažujme funkci jako je na obrázku Obrázek 2. Předpokládá se, že jas pixelu se mezi snímky nemění. K výpočtu vektoru v se použije Newtonova metoda, je to iterativní algoritmus, který velmi rychle konverguje k nule

(většinou stačí 5 iterací). Malá nevýhoda je, že je nutné provést počáteční odhad a pokud tento odhad nebude dostatečně blízký, je možné, že metoda bude divergovat.

To by bylo řešení pro 1D funkci, nyní se budeme zabývat skutečným 2D obrazem. Do vektoru optického toku přibude složka souřadnice y. Potom tedy rovnice optického toku bude mít tvar [9] :

$$I_x u + I_y v + I_t = 0 \qquad (3.6)$$

Bohužel tím přibude další neznámá, nikoliv však rovnice. Máme tedy jednu rovnici a dvě neznámé. Při výpočtu optického toku se vyskytne aperturní problém. Derivační okénko je malé a významný bod se nepodobá rohu, ale spíše hraně. V tomto případě nelze s určitostí říci, k jakému pohybu došlo.



Obrázek 3: Znázornění aperturního problému

Tento problém lze řešit za předpokladu, že okolní body kolem významného bodu se nacházejí ve stejné rovině a vykonávají stejný pohyb jako sám centrální bod [9]. Například při uvažování okénka 5*5 kolem významného bodu vznikne 25 rovnic.

$$\underbrace{\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_{25}) & I_y(p_{25}) \end{bmatrix}}_{A\ 25\times 2} \underbrace{\begin{bmatrix} u \\ v \end{bmatrix}}_{d\ 2\times 1} = -\underbrace{\begin{bmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_{25}) \end{bmatrix}}_{b\ 2\times 1} \tag{3.7}$$

Tento systém rovnic je již přeurčený, lze ho řešit metodou nejmenších čtverců ve standardní formě tak, že minimalizujeme funkci $\|Ad-b\|^2$, potom platí:

$$\underbrace{\left(A^T A\right)}_{2x2}\underbrace{d}_{2x1} = \underbrace{A^T b}_{2x2} \tag{3.8}$$

To lze rozepsat jako [9] :

$$\underbrace{\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}}_{A^T A} \begin{bmatrix} u \\ v \end{bmatrix} = -\underbrace{\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}}_{A^T b} \tag{3.9}$$

A řešením této rovnice je :

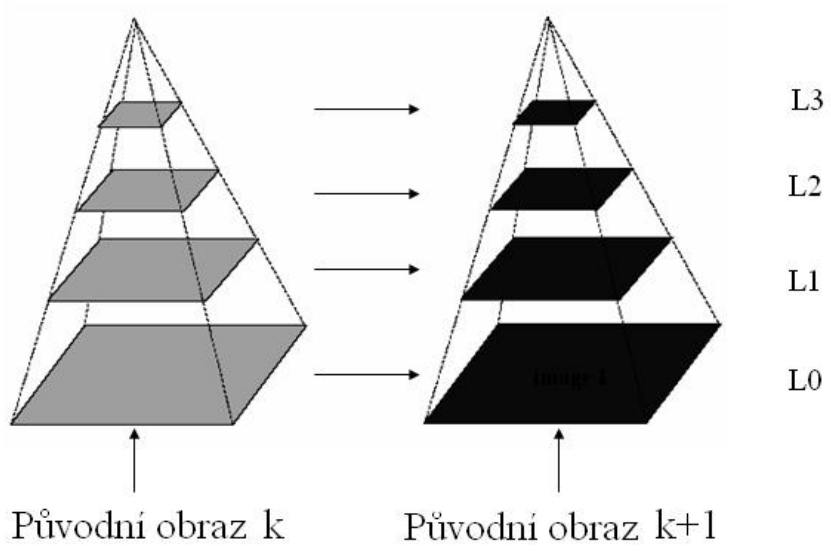
$$\begin{bmatrix} u \\ v \end{bmatrix} = (A^T A)^{-1} A^T b \tag{3.10}$$

Tato rovnice má řešení pokud je A^TA invertibilní, to je za předpokladu, že je matice A^TA plného řádu, to znamená řádu dva, a to platí, pokud se v ní vyskytují dvě velká vlastní čísla. A jsme opět u diskuze o vhodnosti bodů k trasování. Jak bylo rozebíráno v kapitole VYhledání výrazných bodů v obraze, velká vlastní čísla naznačují velkou derivaci ve dvou na sebe kolmých směrech (pravděpodobně roh). Čili tyto body jsou velmi vhodné k trasování a uvedený algoritmus si s nimi dobře poradí.

4.2 PYRAMIDOVÁ REPREZENTACE OBRAZU

Vstupní obraz o velikosti *nx* x *ny* označím jako $\mathbf{I}^0$ - to odpovídá vrstvě nula. Počítání pyramid probíhá rekurzivně, tedy $\mathbf{I}^1$ se počítá z $\mathbf{I}^0$, $\mathbf{I}^2$ z $\mathbf{I}^1$, a tak dále. Potom platí [4] :

$$n_x^L \le \frac{n_x^{L-1}+1}{2}$$
$$n_y^L \le \frac{n_y^{L-1}+1}{2} \qquad (3.11, 3.12)$$



Obrázek 4: Pyramidové zpracování obrazu

Hloubka pyramidy se určuje heuristicky, většinou nabývá hodnot 2,3 nebo 4. Pro většinu aplikací nemá smysl počítat s vyšší úrovní než 4. Dáno velikostí vstupního obrázku. Například pro vstupní obraz velikosti 640 x 480, mají subpyramidy následující velikosti (v pixelech) – 320 x 240, 160 x 120, 80 x 60 a 40 x 30. Počítat další úroveň postrádá smysl. Při použití pyramid je možné zachytit větší posun pixelu než je integrační okno velikosti ω_x a ω_y, proto by se měla hloubka pyramidy volit s ohledem na předpokládanou velikost vektoru optického toku. Nakonec se použije nízkofrekvenční filtr [¼ ½ ¼] x [¼ ½ ¼]^T z důvodu anti-aliasingu před provedením podresamplování obrazu.

4.3 PYRAMIDOVÁ IMPLEMENTACE TRASOVACÍHO ALGORITMU

Každé $L \in \langle 0, L\max \rangle$ stanoví $u^L = \begin{bmatrix} u_x^L & u_y^L \end{bmatrix}$, to jsou souřadnice bodu u v pyramidovém obrázku. Podle předchozích odvození se vektor $\boldsymbol{u^L}$ vypočítá nezávisle v obou osách jako :

$$u^L = \frac{u}{2^L} \tag{3.13}$$

Samotný algoritmus začíná výpočtem optického toku pro vrstvu pyramidy s nejvyšším indexem, tj. L_{max}. Poté se výsledek přesune o úroveň níže jako počáteční odhad pro výpočet „kultivovaného optického toku“ (refined optical flow). To se provádí, dokud se nedojde do úrovně L_0, což je obrázek v původní velikosti.
Podívejme se blíže na matematické operace prováděné při jednom cyklu výpočtu mezi úrovněmi L+1 a L. Předpokládejme, že počáteční odhad optického toku pro úroveň L, kde $g^L = \begin{bmatrix} g_x^L & g_y^L \end{bmatrix}$ je k dispozici z předchozího výpočtu pro úroveň L+1. Potom pro výpočet optického toku pro úroveň L je nutné najít rozdílový pohybový vektor $d^L = \begin{bmatrix} d_x^L & d_y^L \end{bmatrix}^T$, při minimální chybové funkci ε^L.

$$\varepsilon^L(d^L) = \varepsilon(d_x^L, d_y^L) = \sum_{x=u_x^L-\omega_x}^{u_x^L+\omega_x} \sum_{y=u_y^L-\omega_y}^{u_y^L+\omega_y} (I^L(x,y) - J^L(x+g_x^L+d_x^L, y+g_y^L+d_y^L))^2 \tag{3.14}$$

Všimněte si, že integrační okno zůstává pořád stejně velké pro všechny úrovně L. Počáteční odhad, vektor $\boldsymbol{g^L}$, je využit k přiblížení hledaného obrazového místa v obraze J. Tím pádem je velikost rozdílového optického toku malá a proto lehce počitatelná metodou Lucas-Kanade. Výsledek výpočtu se přenese do další úrovně a vytvoří se z něj další počáteční odhad [4] :

$$g^{L-1} = 2\left(g^L + d^L\right) \tag{3.15}$$

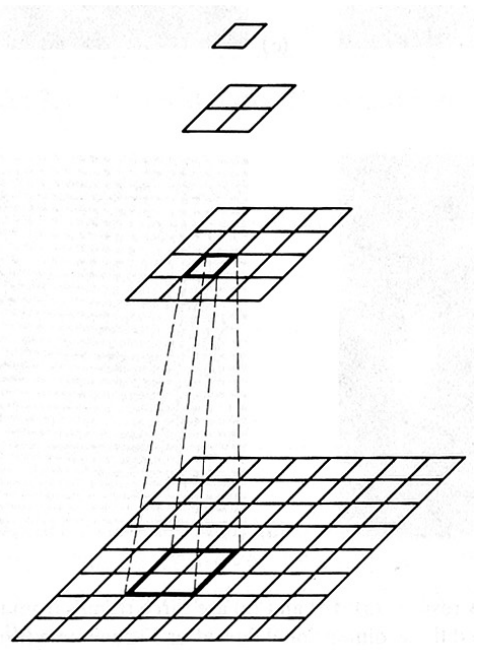
Výpočet rozdílového optického toku se opakuje pro další zbylé úrovně, dokud se nepropracuje k originálnímu obrázku. Výsledný optický tok je tedy vypočten jako:

$$d = g^0 + d^0 \tag{3.16}$$

Lze zapsat i jako:

$$d = \sum_{L=0}^{L_m} 2^L d^L \tag{3.17}$$

Jasná výhoda pyramidové implementace je, že každý rozdílový optický tok (vektor $\boldsymbol{d}^{\boldsymbol{L}}$) může být velmi malý i přesto, že výsledný rozdílový vektor $\boldsymbol{d}$ bude velký. Každý krok výpočtu optického toku je omezen maximální velikostí pohybu pixelu d_{max}. Potom výsledná maximální velikost pohybu pixelu zjistitelná při pyramidové implementaci je $d_{celk.\max} = (2^{L_m+1} - 1)d_{\max}$. Například pro hloubku $L_m = 3$, je výsledná maximální změna pixelu 15x větší než by odpovídalo použitému integračnímu oknu.



Obrázek 5: Naznačení principu pyramidové implementace

4.4 JAK ALGORITMUS FUNGUJE MATEMATICKY

Jak ukázali předchozí rovnice, cílem je najít vektor $\bar{\nu}$ s podmínkou minimální velikosti chybové funkce $\varepsilon(\bar{\nu})$. Iterativní index označíme jako k, kde

$k \geq 1$. Algoritmus se volá rekurzivně a při výpočtu se využívá odhad výpočtu z předchozího kroku. Tím je vlastně popsána rekurzivní metoda nejmenších čtverců:

$$\overline{\nu^{k-1}} = \left[\nu_x^{\,k-1} \quad \nu_y^{\,k-1}\right]^T \tag{3.18}$$

Proto:

$$B_k(x,y) = B(x + \nu_x^{\,k-1}, y + \nu_y^{\,k-1}) \tag{3.19}$$

$$\varepsilon(\overline{\eta^k}) = \varepsilon(\eta_x^k, \eta_y^k) = \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} (A(x,y) - B_k(x+\eta_x^k, y+\eta_y^k))^2 \tag{3.20}$$

$$\overline{\eta^k} = G^{-1} \cdot \overline{b_k} \tag{3.21}$$

Dalšími úpravami se dojde až k tvaru:

$$\overline{b}_k \cong \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \begin{bmatrix} \delta I_k(x,y) \cdot I_x(x,y) \\ \delta I_k(x,y) \cdot I_y(x,y) \end{bmatrix} \tag{3.22}$$

$$\delta I_k(x,y) = A(x,y) - B_k(x,y) \tag{3.23}$$

Všimněme si, že prostorové derivace $\boldsymbol{I_x}$ a $\boldsymbol{I_y}$ (ve všech směrech v okolí bodu p) se vypočítají pouze jednou na začátku. Proto matice $\boldsymbol{G}$, zůstává během výpočtu optického toku pro danou úroveň stejná [4]. Tím se snižuje počet prováděných výpočtů. Jediný parametr, který je nutný počítat v každém kroku je vektor b_k, který zachycuje zbylé rozdíly mezi obrazem a přiloženou maskou po posunu vektoru $\overline{\nu^{k-1}}$. Pro první iteraci se použije počátečný odhad :

$$\overline{\nu^0} = [0 \quad 0]^T \tag{3.24}$$

Potom je počítán pomocí předchozího odhadu v iteračním cyklu:

$$\overline{\nu^k} = \overline{\nu^{k-1}} + \overline{\eta^k} \tag{3.25}$$

Iterační cyklus pokračuje, dokud není dosaženo menšího přírůstku než je zadaný práh ($\overline{\eta^k}$ <T), nebo dokud není dosaženo maximálního počtu iterací. Zpravidla již pátá iterace bývá úspěšná, ovšem záleží na rozmanitosti scény.

Dejme tomu, že pro dosažení konvergence je nutné provést k iterací, potom výsledné řešení optického toku bude mít tvar:

$$\bar{v} = d^{L} = \bar{v}^{K} = \sum_{K=1}^{K} \bar{\eta}^{K} \tag{3.26}$$

Tento vektor minimalizuje velikost chybové funkce. Na konci provedeného počtu iterací optického toku se zjištěný vektor d^L dále upřesňuje pomocí pyramidové implementace. A úplně na závěr se vypočítá poloha bodu v obraze J podle základního vztahu optického toku:

$$v = u + d \tag{3.27}$$

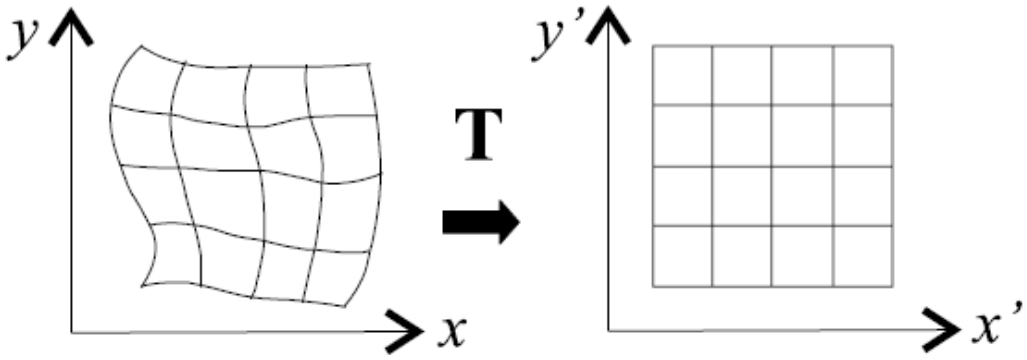
Algoritmus Lucas-Kanade je rovněž obsažen v knihovně OpenCV (viz kapitola 6), kde je algoritmus vhodně implementován po funkční i výkonové stránce.

4.5 OPTIMALIZACE ALGORITMU

Vzhledem k tomu, že trasování pohybu probíhá z videosekvence, je vhodné udělat několik úprav. Algoritmus využívá pyramidové implementace, počítání jednotlivých pyramid obrazu je výpočetně náročné, je to několikeré resamplování obrázku (podle počtu pyramid). Pokud už je obrazová pyramida jednou vypočítána, není nutné ji v následujícím kroku počítat znova, z následujícího obrázku se stává aktuální a může si vypočtenou pyramidu přenést sebou. Dále (jak bylo naznačeno v části 4.1) při výpočtu optického toku hraje velkou roli počáteční odhad, čím je lepší tím je větší pravděpodobnost rychlejší konvergence. Sekvenční zpracování snímků poskytuje velmi dobré předpoklady k odhadu dalšího pohybu kamery. Použitím těchto úprav může dojít k podstatnému zrychlení programu.

5. GEOMETRICKÉ TRANSFORMACE

Používají se pro zvětšování, posouvání, pootáčení, zkosení obrazů. V oblasti počítačového vidění je motivace využití geometrických transformací k odstranění geometrických zkreslení.



Obrázek 6: Příklad geometrické transformace v 2D

Rozepíšeme vektorovou transformaci T do dvou složek [11]

$$x' = T(x) \quad \rightarrow \quad x' = T_x(x, y), \qquad y' = T_y(x, y) \qquad (5.1), (5.2), (5.3)$$

Při transformaci souřadnic bodů se počítá poloha každého bodu ve spojitých souřadnicích.

5.1 POPIS ZÁKLADNÍCH TRANSFORMACÍ

Posun:	P’ = P + T	(5.4)
Rotace:	P’ = P * R	(5.5)
Měřítko:	P’ = P * S	(5.6)
Zkosení:	P’ = P * S_H	(5.7)

Bohužel je posun na rozdíl od ostatních operací vyjádřen součtem, oproti součinu u ostatních operací, proto je tento popis nevýhodný. Aby bylo možno k těmto transformacím přistupovat jednotně, zavedl se popis s homogenními souřadnicemi P(x,y,1).Při použití homogenních souřadnic lze základní geometrické transformace popsat maticově pomocí tzv. Matice homogenní transformace.

5.2 MATICE HOMOGENNÍ TRANSFORMACE PRO ZÁKLADNÍ 2D TRANSFORMACE

Popis základních transformací v afinním prostoru, bylo využito obrázků z materiálů [11],[12].

Obecná transformační matice:

R – matice rotace

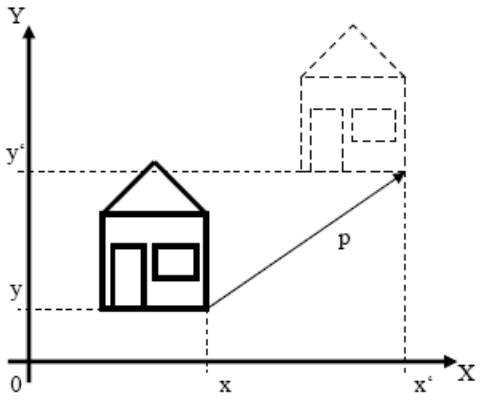
T – translační složka

P – perspektiva;

S – měřítko

$$\begin{bmatrix} R & R & T_x \\ R & R & T_y \\ P_x & P_y & S \end{bmatrix} \qquad (5.8)$$

1.Posunutí (translace)

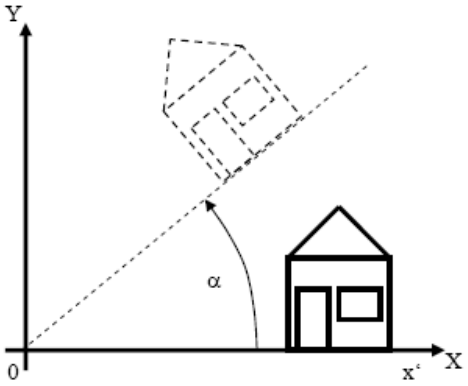


$$H_T = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \qquad (5.9)$$

$$x' = x + t_x$$
$$y' = y + t_y \qquad (5.10)$$

Obrázek 7: Posun v obraze

2. Rotace



Obrázek 8: Ukázka rotace obrazu

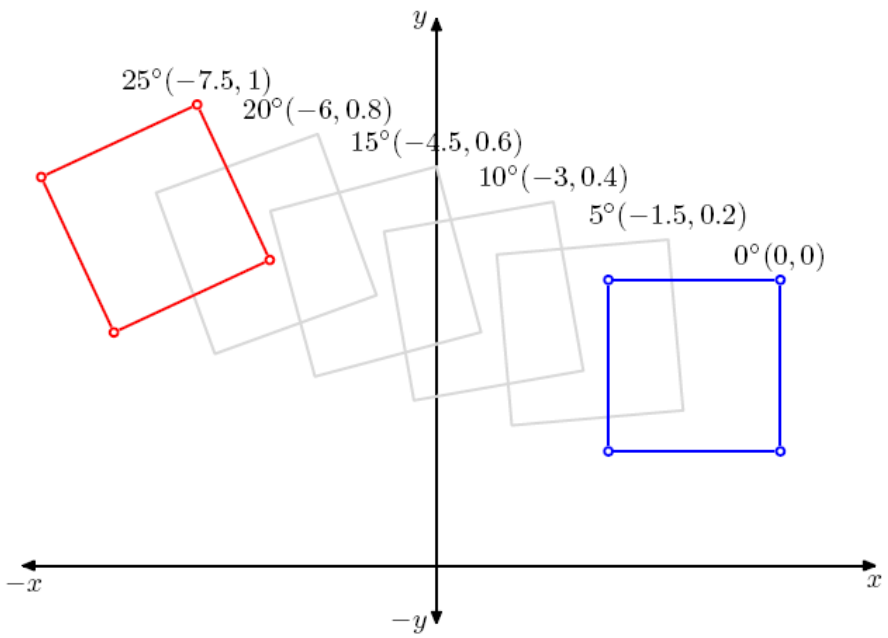
Matice popisující rotaci:

$$H_R = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (5.11)$$

x' = x*cos α – y*sin α

y' = x*sin α + y*cos α (5.12)

3.Kombinace translace a rotace



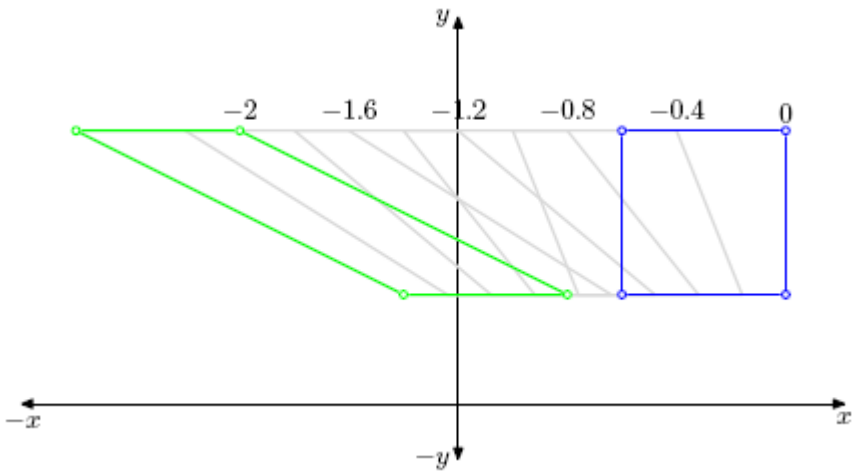
Obrázek 9: Kombinace translace a rotace v obraze

x' = x*cos α – y*sin α + t_x

y' = x*sin α + y*cos α + t_y (5.13)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha & t_x \\ \sin\alpha & \cos\alpha & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (5.14)$$

4.Zkosení v ose x



Obrázek 10: Zkosení objektu v ose x

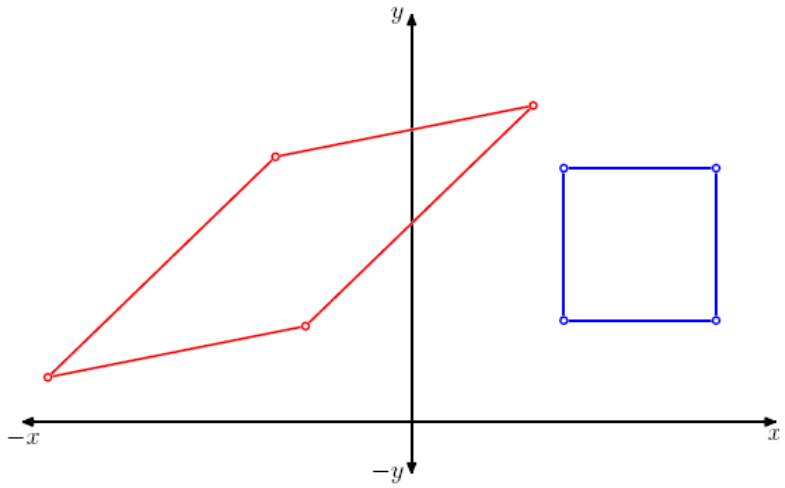
x’ = x+ay

y’ = y (5.15)

Obdobně se definuje zkosení v ose y

5.Afinní transformace

Sloučením všech předchozích transformací vznikne afinní transformace [11].



Obrázek 11: Afinní transformace

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} R_{xx} & R_{xy} & t_x \\ R_{yx} & R_{yy} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{5.16}$$

$$\begin{aligned} x' &= R_{xx}x + R_{xy}y + t_x \\ y' &= R_{yx}x + R_{yy}y + t_y \end{aligned} \tag{5.17}$$

5.3 ZAVEDENÍ 2D PROJEKTIVNÍ TRASFORMACE

Projektivita je takové invertibilní zobrazení $h : P^2 \longrightarrow P^2$, pro které platí, že tři body x_1,x_2,x_3 leží na té samé přímce tehdy, jestliže $h(x_1),h(x_2),h(x_3)$ také. 2Dprojektivní transformace je lineární transformace tří dimenzionálních homogenních vektorů reprezentována regulární 3x3 maticí [11].

$$\begin{bmatrix} x_1' \\ x_2' \\ x_3' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{5.18}$$

Zjednodušeně lze zapsat jako:

$$x' = \mathbf{H}x \tag{5.19}$$

Matice **H** se nazývá matice homogenní transformace.

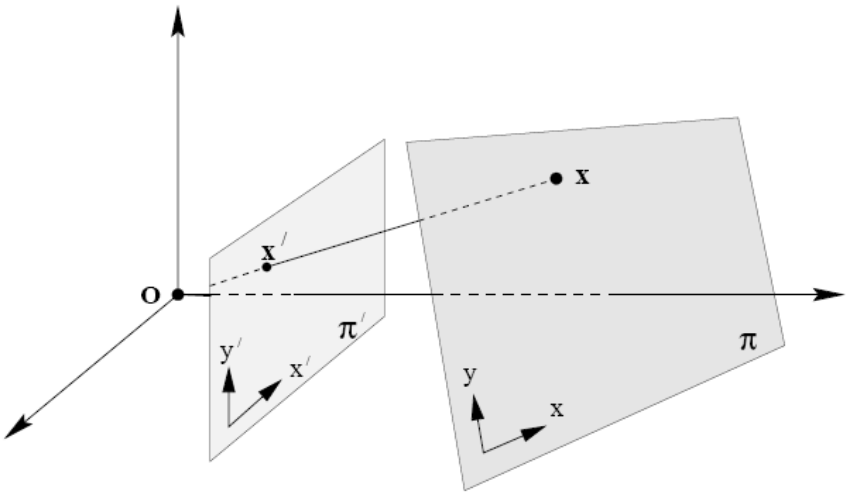
Rovněž platí, že :

$$\begin{bmatrix} x_1' \\ x_2' \\ x_3' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} h_1^T \\ h_2^T \\ h_3^T \end{bmatrix} x \tag{5.20}$$

Měření probíhá v $(x,y)^T=(x_1/x_3,x_2/x_3)$, obdobně pro výsledek platí $(x',y')^T=(x'_1/x'_3,x'_2/x'_3)^T$. Nejvhodnější volba x_3 je $x_3=1$ [11].

Potom :

$$\begin{aligned} x' &= \frac{x'_1}{x'_2} = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}} = \frac{h_1^T x}{h_3^T x} \\ y' &= \frac{x'_2}{x'_3} = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}} = \frac{h_2^T x}{h_3^T x} \end{aligned} \tag{5.21}$$



Obrázek 12: Zobrazení z roviny do roviny, středová projekce

Po úpravě [11] :

$$\begin{aligned} x'\left(h_{31}x+h_{32}y+h_{33}\right)&=h_{11}x+h_{12}y+h_{13}\\ y'\left(h_{31}x+h_{32}y+h_{33}\right)&=h_{21}x+h_{22}y+h_{23}\end{aligned} \tag{5.22}$$

Pro každý korespondující pár $(x_i,y_i)^T \leftrightarrow (x'_i,y'_i)^T$ platí:

$$\begin{aligned} x'\left(h_{31}x+h_{32}y+h_{33}\right)-\left(h_{11}x+h_{12}y+h_{13}\right)&=0\\ y'\left(h_{31}x+h_{32}y+h_{33}\right)-\left(h_{21}x+h_{22}y+h_{23}\right)&=0\end{aligned} \tag{5.23}$$

A totéž zapsané maticově:

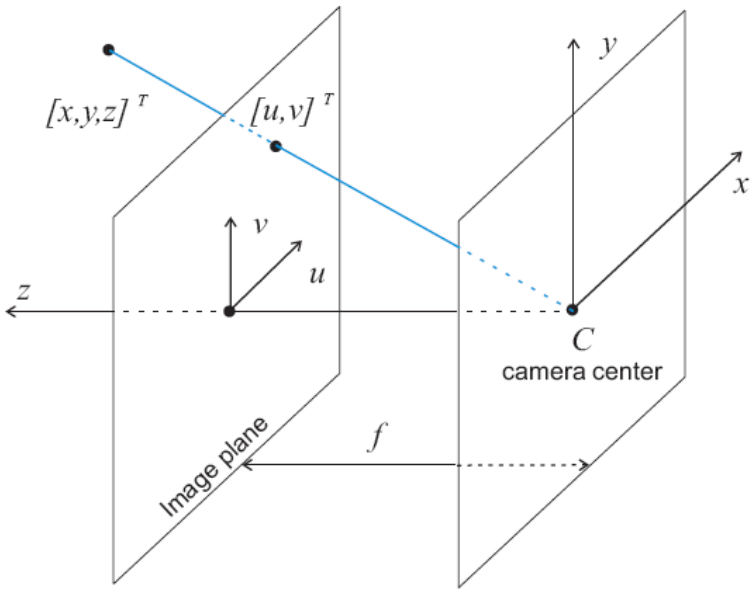
$$\begin{bmatrix} -x_1 & -y_1 & -1 & 0 & 0 & 0 & x_1'x_1 & x_1'y_1 & x_1' \\ 0 & 0 & 0 & -x_1 & -y_1 & -1 & y_1'x_1 & y_1'x_1 & y_1' \\ -x_2 & -y_2 & -1 & 0 & 0 & 0 & x_2'x_2 & x_2'x_2 & x_2' \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -x_n & -y_n & -1 & 0 & 0 & 0 & x_n'x_n & x_n'x_n & x_n'x_n \\ 0 & 0 & 0 & -x_n & -x_n & -1 & y_n'x_n & y_n'x_n & y_n'x_n \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \tag{5.24}$$

V této rovnici se vyskytuje 9 neznámých, ale proměnná h_{33} je závislá na ostatních proměnných, proto pro řešení rovnice stačí výpočet 8 proměnných, jinak řečeno pro řešení stačí 4 korespondující body. Pro méně bodů nelze vypočítat jednoznačné řešení. Vzhledem k tomu, že výpočet pouze pro 4 korespondující body je silně závislý na přesnosti určení korespondencí a jejich výpočet není jednoduchý, vždy se

v něm uplatňuje určitá nepřesnost a navíc je podstatná volba samotných bodů. Výhodnější je počet korespondujících bodů ještě více rozšířit, tím vznikne přeurčený systém rovnic, který se následně řeší pomocí metody nejmenších čtverců.

5.4 MODEL KAMERY

Nejjednodušším modelem kamery, přesto pro svoji jednoduchost v praxi často používaným, je dírková kamera (pinhole). V tomto jednoduchém případě vstupuje ze scény do kamery jediný paprsek od každého objektu skrz jeden bod. U fyzikálního modelu dírkové kamery se tento bod zobrazuje na snímací plochu kamery. Vzniklá plocha obrázku se většinou nazývá projektivní plocha, leží vždy v ohnisku a výsledná velikost obrázku vztaženého ke vzdáleným objektům je dána parametrem kamery, který se nazývá ohnisková vzdálenost. Pro ideální dírkovou kameru je vzdálenost mezi průzorem kamery a scénou přesně ohnisková vzdálenost, jak ukazuje obrázek Obrázek 13 převzato z [11].



Obrázek 13: Model dírkové kamery (pinole camera)

Promítání objektu:

$$u = f\frac{x}{z}$$
$$v = f\frac{y}{z} \qquad (5.25), (5.26)$$

f – ohnisková vzdálenost, Proměnné f, x, y jsou lineární, z je nelineární

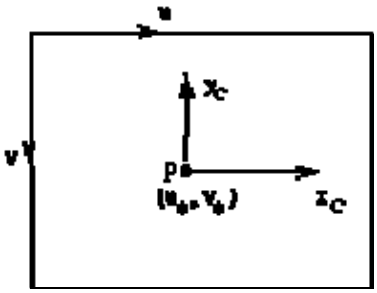
Způsob, jakým se skutečné souřadnice (*x,y,z*) promítnou na projektivní plochu jako souřadnice (x_c,y_c), se nazývá projektivní transformace. Při práci s touto transformací je vhodné používat homogenní souřadnice. Tyto souřadnice se přidružují s bodem dimenze projektivního prostoru báze *n* jako (*n+1*) dimenzionální vektor. Například pro souřadnice bodu *x,y* jsou homogenní souřadnice *x,y,w* s dodatečnou podmínkou, že všechny body s proporcionálními hodnotami si odpovídají.

Vyjádření lineárního mapování:

$$\begin{bmatrix} x_c \\ y_c \\ f \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad (5.27)$$

5.5 ZOBRAZOVACÍ GEOMETRIE

Pro zjednodušení lze přenést skutečné souřadnice objektu (x,y) na souřadnice (u,v) [11].



Obrázek 14: Zobrazovací geometrie kamery

Poznámka: u_0,v_0 je průsečík optické osy a projekční plochy

Promítnutí bodů v kameře:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} fk_u & 0 & u_0 \\ 0 & fk_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ f \end{bmatrix} = \mathbf{C} \begin{bmatrix} x_c \\ y_c \\ f \end{bmatrix} \qquad (5.28)$$

Kde **C** je kalibrační matice kamery

V euklidovské geometrii se přesný model pohybu kamery popisuje pomocí [12]:

Xc = R Xw + T

Aplikací tohoto modelu se změní předchozí rovnice na:

$$\mathbf{x} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{C} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R}^{\top} & \mathbf{t} \\ \mathbf{0}_3^{\top} & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{5.29}$$

Potom se tedy projekční matice P z 3-dimenzionálního euklidovského prostoru do obrazové roviny zapíše jako matice 3x4:

$$P = C[R \mid T] \tag{5.30}$$

Zvolením souřadnice Z v objektovém prostoru rovno nula, dojde ke zjednodušení [11]:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{12} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{14} \\ p_{21} & p_{12} & p_{24} \\ p_{31} & p_{32} & p_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{5.31}$$

Čili tím vznikne transformace nazývaná homografie:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \sim \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{12} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{5.32}$$

Pro zadanou úlohu je použití modelu dírkové kamery dostačující.

6. OPEN COMPUTER VISION

6.1 POPIS KNIHOVNY OPEN CV

Knihovna Open Source Computer Vision Library od Intelu® (zkráceně OpenCV), která jak už z názvu vyplývá, je knihovna bezplatná a to jak pro nekomerční, tak i komerční využití. Navíc je multiplatformní, využitelná pod různými OS (Windows,Linux, Mac OS). Oficiální stránky Open CV [5]. Možnost stažení vlastní knihovny a technické dokumentace je na Sourceforge [6].

OpenCV poskytuje průhledné a jednoduché rozhraní k využití IPP Intel® Integrated Performance Primitives [7]. Jedná se o rutiny a knihovní funkce, obsahující velmi optimalizované softwarové funkce pro zpracování dat a multimédií, umožňující využít více jádrové procesory. Knihovna je psána v C/C++ a přesto, že je vyvíjena celou řadou autorů, existují přesné pokyny jak funkce v OpenCV psát. Knihovna OpenCV není vázána na OS systém, dokonce ani na hardware (respektive existují upravené verze téměř pro cokoli). V současné době skupina lidí provádí konverzi (optimalizaci) knihoven pro Playstation 3 od SONY, jde o herní konzoli, u které je na výbornou grafiku, tudíž rychlé zpracování kladen důraz. Vzhledem k jejich masové výrobě jde o poměrně levné zařízení (grafické čipy) – po optimalizaci bylo dosaženo u většiny funkcí větší rychlosti zpracování než u srovnatelně výkonného PC.

OpenCV umožňuje provádět vysokoúrovňové optimalizované operace vhodné pro počítačové vidění, také se snaží poskytovat prostředky pro vývoj aplikací běžících v reálném čase. Cílem je pomáhat komerčnímu využití počítačového vidění a jeho využití v robotice, bioinformatice, zpracování signálů, monitoringu a bezpečnosti. Poskytováním otevřené infrastruktury, vytváří dobré zázemí pro vytváření celosvětových vývojářských komunit, které mohou spolupracovat při vývoji ucelených a velmi výkonných aplikací.

Součástí OpenCV je rovněž API, které umožňuje přímo pomocí funkcí OpenCV provádět základní operace s obrázky a videem jako je načítání, ukládání a zobrazování pro různé datové typy. Například při načtení obrázku stačí zadat funkci

pouze jméno obrázku a funkce si sama podle přípony zjistí datový typ a podle něj provede načtení obrázku.

Oficiálně podporovaná diskusní skupina se nachází na serveru Yahoogroups. Lze zde najít mnoho návodů, nápadů a nalezených chyb či různých vylepšení. Je to velmi silná a stále se rozvíjející komunita (aktuální počet více než 30000 registrovaných členů).

6.2 MODULY OPENCV

OpenCV se skládá z 5 základních modulů :

1. **CXCORE** – obsahuje definice základních statických a dynamických struktur a funkcí pro vykreslování
2. **CV** - knihovna počítačového vidění - obsahuje hranové operátory, filtry, transformace, morfologické operace, apod.
3. **HighGUI** – zobrazování a ukládání obrázků, práce s videem, interakce (callback funkce)
4. **CVCAM** – zpracování a ovládání toku digitálního videa, nastavení kamery, kalibrace kamery
5. **Machine learning** – pokročilé metody, umělá inteligence

Hlavní výhodou OpenCV je vysoká výkonnost. Té bylo dosaženo hlavně díky použití vlastních datových typů, myšleno na nejnižší úrovni deklarace datových typů. Standardní datové typy používané v C++ lze použít, ale většinou je výhodnější z důvodu postupného navazování funkcí využívat datové typy zavedené v OpenCV, navíc většina těchto datových typů a struktur je vhodně optimalizována a tím se lépe využije výkon procesoru. OpenCV umožňuje využívání již vytvořených funkcí, ale zároveň obsahuje velmi účinné nástroje pro vytvoření vlastních struktur, funkcí, procedur a filtrů. Zároveň u většiny struktur existuje více způsobů, jak se dostat k vlastním uloženým datům, od ukazatelového přístupu až po hromadné načítání/vkládání bloků dat. Díky dobře navrženému rozhraní je práce s OpenCV velmi transparentní, dostupná dokumentace je na velmi vysoké úrovni a rovněž velmi široká komunita lidí využívajících OpenCV poskytuje mnoho užitečných rad.

7. PRAKTICKÁ REALIZACE

7.1 APLIKACE CAMERATRACER

Pro vyzkoušení navrženého algoritmu byla vytvořena testovací aplikace CameraTracer v C++ využívající knihovny OpenCV. Pro zjednodušení testovací fáze byla k programu přidána nástavba v podobě jednoduchého grafického rozhraní pro operační systém Windows, konkrétně WIN API.

Aplikace CameraTracer zpracovává jednotlivě vstupní dvojice snímků, najde výrazné body v prvním snímku pomocí upravené metody GoodFeatureToTrack (funkce popsána v kapitole 2.2, základem je funkce z OpenCV). Dále pomocí algoritmu optického toku (popsaného v kapitole 4, implementace pomocí OpenCV) vypočítá optický tok pro vstupní výrazné body. Následně jsou tyto výsledky zpracovány, aby se zjistila relace mezi snímky. Nyní podrobně o jednotlivých částech aplikace.

7.2 VSTUPNÍ DATA

Vstupem do programu je dvojice snímků získaná z videosekvence. Původně jsem zamýšlel použít jako vstup video formátu AVI, ale to se ukázalo jako nevhodné, z důvodu nepřímého přístupu ke vstupním obrázkům a stálého zatěžování počítače dekomprimováním těch samých obrázků. Daleko vhodnější je vytvoření indexovaného seznamu obrázků. Pro tento účel jsem si vytvořil pomocný program AviToJpg. Ten z načteného videa, pro zadaný počet snímku, vytvoří sekvenci obrázků. Tento program je vytvořen rovněž v C++ s využitím OpenCV (viz kapitola 9).

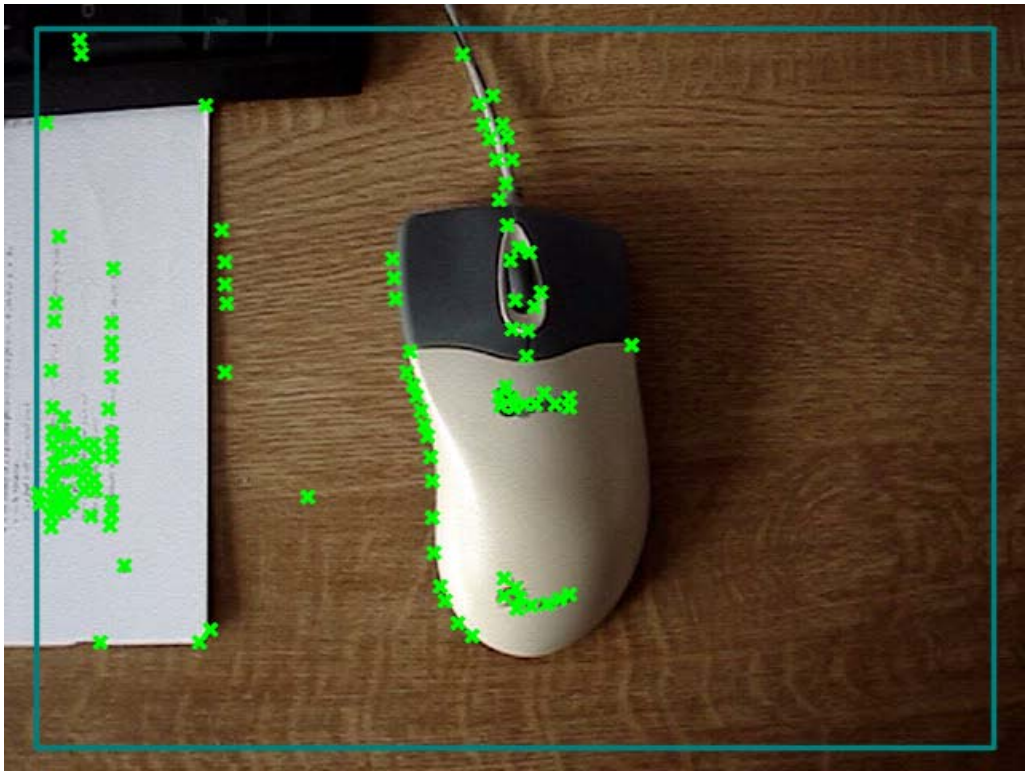
7.3 VYHLEDÁVÁNÍ VÝZNAMNÝCH BODŮ

Jak již bylo zmíněno, byl využit algoritmus GoodFeatureToTrack (viz kapitola 2.2).Tento algoritmus vyhledá vhodné body velmi spolehlivě, ale tím, že hledá místa největšího gradientu jasové funkce, preferuje světlejší místa a různorodé objekty. Může nastat situace kdy jeden výrazný objekt nebo více osvícená část scény si pro sebe „posbírá“ všechny výrazné body, tato situace je z globálního hlediska

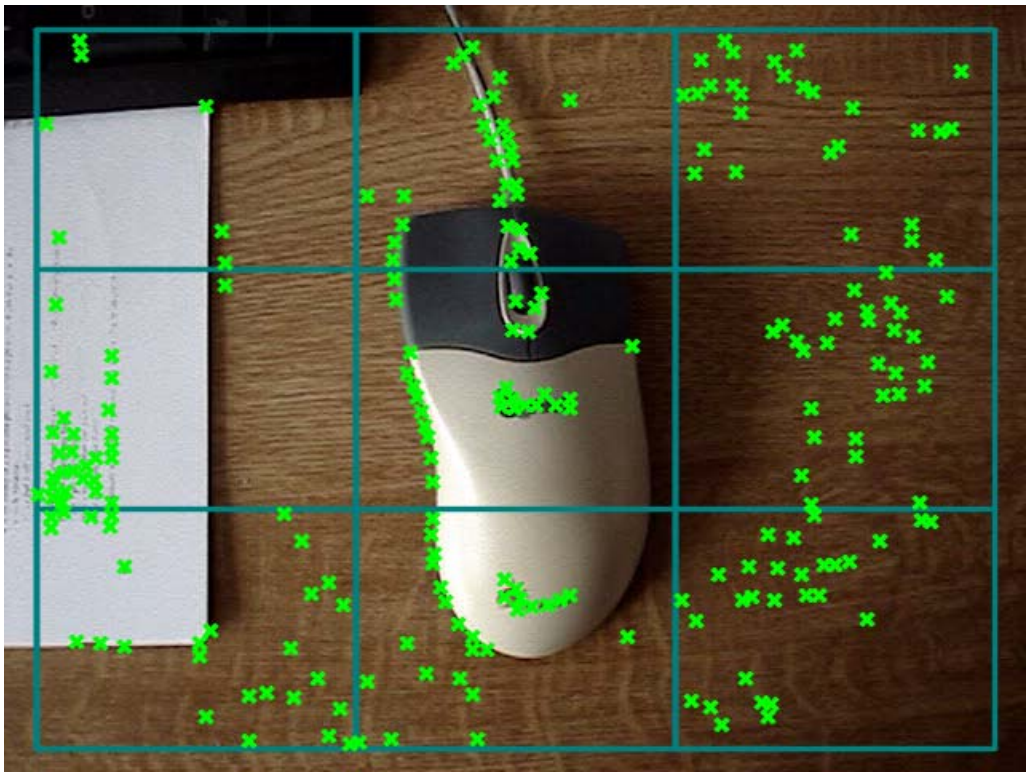
nevhodná. Pro trasování pohybu kamery je výhodnější, když jsou významné body v obraze více rozprostřeny a během trasování se výrazně nepřesouvají po snímku. Na druhou stranu je potom obtížnější v některých oblastech dostatečně spolehlivě určit optický tok, protože body které obsahují, nejsou tak výrazné. To lze řešit pomocí filtrace získaných odhadů.

7.3.1 Použité rozšíření algoritmu GoodFeaturesToTrack

Aby se zabránilo nadměrnému shlukování významných bodů jen v určitých oblastech, provedl jsem rozdělení snímku na více nezávislých oblastí, ve kterých se významné body hledají. Navíc body příliš blízko hranic snímku se z hlediska trasování jeví jako nevhodné, na následujícím snímku se nemusí vůbec objevit, vypadnou za hranice snímané oblasti. Z těchto důvodů jsem algoritmus poupravil a významné body hledám nezávisle ve více oblastech. Na obrázku Obrázek 16 je ilustrována situace, kde je snímek rozdělen na 9 stejně velkých oblastí, krajních 15px u výšky a 20px u šířky je vynecháno. Hranice jednotlivých oblastí je znázorněna modrou barvou.



Obrázek 15: Výsledek algoritmu good features to track



Obrázek 16: Výsledek upraveného algoritmu good features to track – 9 oblastí

Maximální počet hledaných bodů je omezen na 250 bodů stejně jako na obrázku Obrázek 15, ovšem ve skutečnosti jich je na obrázku Obrázek 16 více, je to ovlivněno parametrem minimální vzdálenost, který způsobuje vynechání bodů ležících příliš blízko sebe.

Jako vhodné řešení se ukázalo rozdělení na 25 bloků. Když vezmu, že vstupní obrázek má velikost 640px na 480px, potom tedy zvolím šířku pole 120 px a výšku 90 px. Zbudou okraje, ve kterých se body vyhledávat nebudou. V každém poli se vyhledávává až 10 významných bodů (ovlivňuje parametr kvality). Ve snímku se tedy označí až 250 bodů. Vzniklá situace je znázorněna na obrázku Obrázek 34: Nalezené významné body v jednotlivých polích.

7.4 VÝPOČET OPTICKÉHO TOKU

V další fázi se provede výpočet optického toku pomocí algoritmu Lucas-Kanade, tak jak jej popisuje kapitola 4. Pro každý významný bod v prvním obrázku se pomocí optického toku pokusí přiřadit vektor optického toku. V každém jednotlivém políčku, tak jak byly vytvořeny předchozím postupem, vznikne

v nejlepším případě tolik vektorů optického toku, kolik bylo v políčku nalezených významných bodů – znázorňuje obrázek Obrázek 35: Optický tok.

7.5 ZPRACOVÁNÍ ZÍSKANÝCH DAT

Následuje filtrace dat - v každém políčku se provede omezení dat na jeden výsledný vektor. Filtrace se provede pomocí mediánu, ten má oproti prostému průměru velkou výhodu, dokáže si poradit se situací, kdy malý počet vektorů optického toku je chybný. Navíc výpočet mediánu z malého počtu dat je velmi rychlý. Po této filtraci (redukci) dat vznikne na snímku tolik rozprostřených vektorů, kolik je políček na snímku. V ilustrovaném případě jde o 25 vektorů (obrázek Obrázek 17: Translační pohyb). Tyto nalezené vektory se použijí pro výpočet homogenní matice.

Ještě než dojde k vlastnímu výpočtu homogenní matice, provede se předzpracování dat. Z 25 vektorů optického toku (obrázek Obrázek 17: Translační pohyb) se vypočte medián podle délky a podle úhlu. Provede se porovnání délky vektoru s mediánem délky, a pokud se neliší více jak o 20% velikosti mediánu a zároveň úhel se neliší více jak o 0.1 rad (přibližně 5,7°), vektor se označí jako „shodný“ (v obrázcích je naznačeno červenou barvou šipky). V opačném případě se označí modrou šipkou. Motivací je vyfiltrovat nepovedené odhady optického toku, pokud se minimálně 15 vektorů z 25 označí červeně (za „shodné“), označí se pohyb za translační (u rotačního pohybu je rozložení délek a úhlů vektorů větší) a dále se počítá pouze s těmito „červenými“ vektory. Vychází se z typické situace, kdy došlo k translačnímu pohybu a u některých vektorů (v některých oblastech) došlo ke špatnému odhadu optického toku. Pozitivním vedlejším přínosem této filtrace je, že pokud se v obraze vyskytne malý pohybující se objekt a převládající pohyb je translační, dojde k vyfiltrování chybného vektoru. V případě, že „červených“ vektorů je méně než 15 z 25 dále se zpracovávají všechny vektory. Běžný pohyb tělesa se většinou skládá z podstatně většího počtu translačních pohybů než rotačních, tím že se provede předzpracování dat, dojde k velkému zlepšení odhadu trajektorie.

7.6 VÝPOČET MATICE HOMOGENNÍ TRANSFORMACE

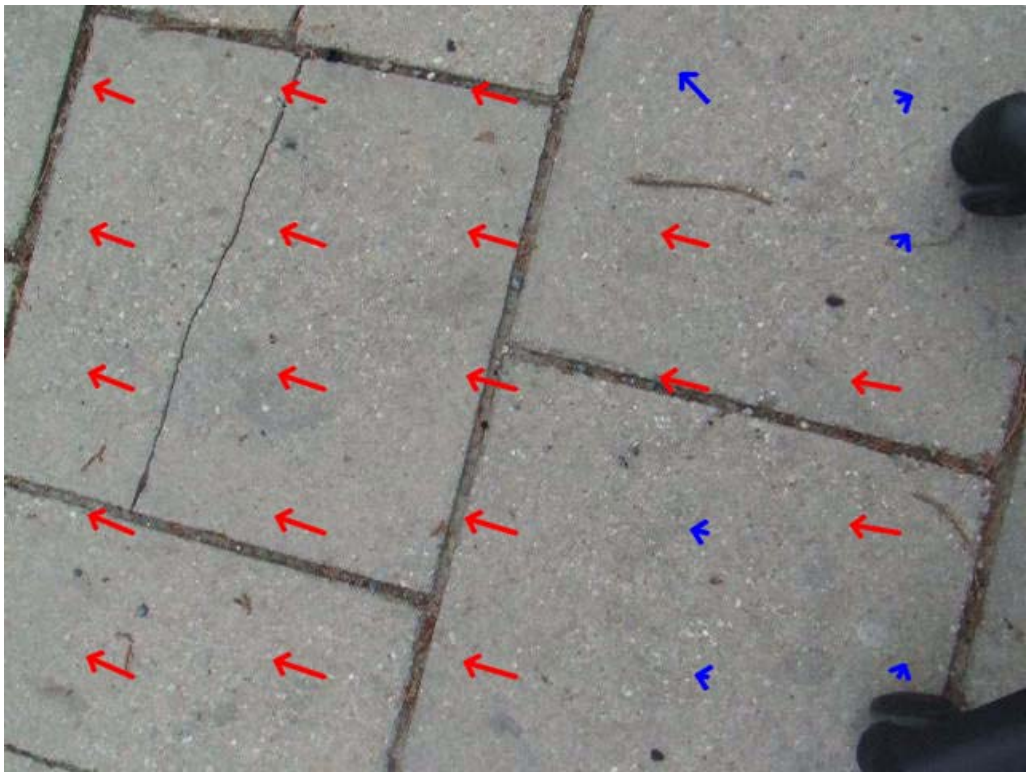
Využitím až 25-ti (záleží na předzpracování, minimální hodnota je 15) korespondenčních bodů se za pomoci metody nejmenších čtverců provede výpočet matice homogenní transformace **H**. K výpočtu je využita funkce z knihovny OpenCV. Při uvažování pohybu kamery ve stále stejné rovině přicházejí v úvahu dva možné pohyby - translační a rotační. Pokud bylo minimálně 15 vektorů označených za shodné (červeně), pohyb je translační a z matice homogenní transformace se odečtou pouze parametry *Tx* a *Ty*. V opačném případě je pohyb označen za rotační a odečítají se jednotlivé složky rotačního pohybu z matice rotace tvořené R_{xx}, R_{xy}, R_{yx}, R_{yy} a počítá se úhel natočení. Výpočet souřadnic při rotačním pohybu:

$$Dx = Tx * cos(\alpha) - Ty * sin(\alpha) \quad (7.1)$$
$$Dy = Tx * sin(\alpha) + Ty * cos(\alpha) \quad (7.2)$$

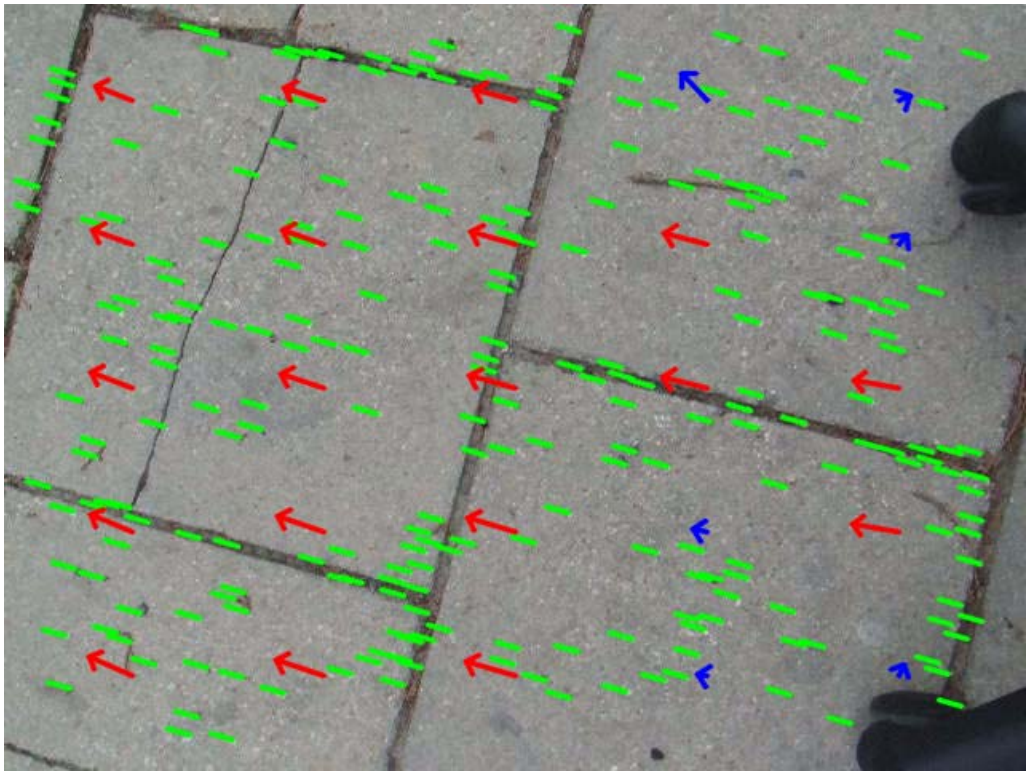
7.7 ZAŘAZENÍ POHYBU A PREDIKCE DALŠÍHO POHYBU

Zjištěné údaje se ukládají, jako trasovací body k tomu slouží struktura TracePoint, ta tvoří základ trasovací mapy - struktura TraceMap. Vzhledem k tomu, že pro výpočet optického toku je interně použita Newtonova iterační metoda, u které je nutná inicializace, a rychlost výpočtu závisí na počátečním odhadu, je vhodné vypočítat predikci pohybu a vhodně nastavit inicializační bod. To je bod, od kterého se začíná hledat poloha význačného bodu v druhém snímku, implicitně je nastavena hodnota rovnající se souřadnicím bodů v prvním snímku.

Pokud je zjištěn translační pohyb, předpokládá se, že pohyb v následujícím snímku bude opět stejným směrem, dáno fyzikálními parametry pohybujícího se objektu zejména hmotností a setrvačností. Proto se souřadnice inicializačního bodu, přesněji pole nalezených význačných bodů, posunou o hodnotu predikce pohybu. Pomocí testování byla vybrána vhodná predikce jako 50% aktuálního pohybu, tím dojde k malému zrychlení výpočtu a většinou i k zlepšení přesnosti odhadu následujícího pohybu. Rovněž se tím zmírní dopad omezení maximálního pohybu mezi jednotlivými snímky. Integrační okno zůstává pořad stejně velké, ale tím že je posunuté blíž k předpokládanému výskytu bodu, je možné nalézt pohyb ve skutečnosti větší, než při neposunutém počátečním odhadu. Přesnost odhadu je zachována.



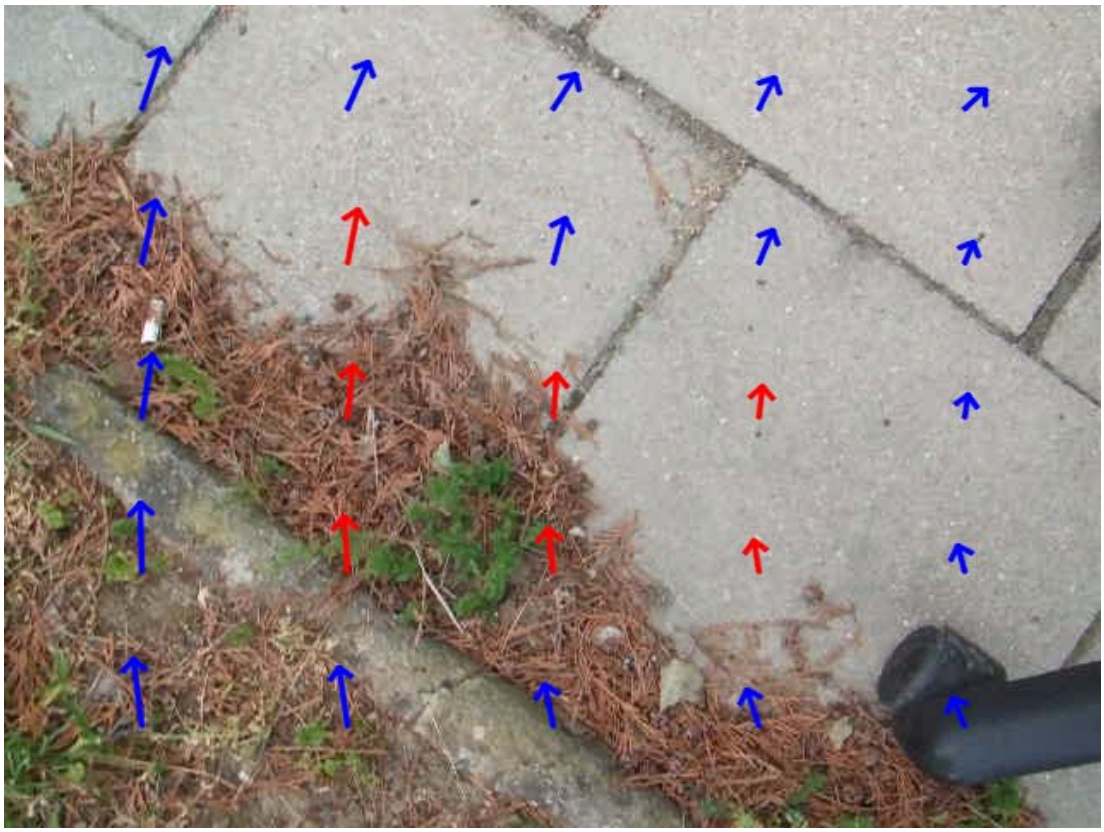
Obrázek 17: Translační pohyb



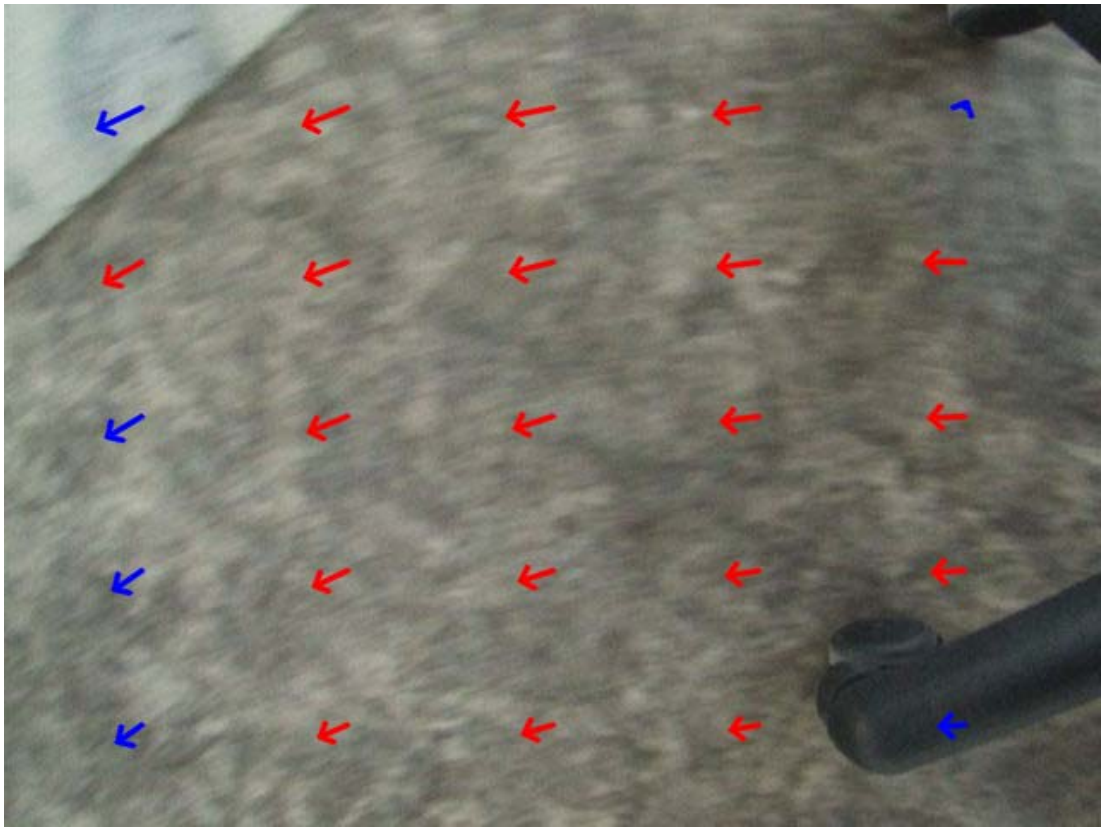
Obrázek 18: Translační pohyb s predikcí následujícího pohybu

Na obrázku Obrázek 17 je znázorněna situace, kdy došlo k translačnímu pohybu. Situace je navíc ztížena tím, že na pravém okraji snímku se nachází kolečka židle, na které byla umístěna kamera a podle očekávání v těchto polích došlo k špatnému odhadu pohybu. Navíc v pravém horním i pravém dolním rohu se v několika polích nezdařil odhad optického toku správně, bylo to ovlivněno velkou homogenitou těchto oblastí.Pomocí popsané mediánové filtrace na úrovni bloků se povede označit tyto špatné odhady a vzhledem k tomu, že většina ostatních vektorů má stejnou velikost i směr, výpočet se provede pouze pomocí „červených“ vektorů. Následně se ještě provede predikce následujícího pohybu - označeno zelenými úsečkami na obrázku Obrázek 18 (úsečky vycházejí z míst označených jako významné body).

Pokud je pohyb označen za rotační predikce pohybu se neprovádí. Hlavním důvodem je, že z homogenní matice nelze jednoznačně vypočítat osu otáčení pro libovolný otáčivý pohyb. Navíc otáčivý pohyb nebývá tak kontinuální jako posuvný pohyb a špatná predikce má kontraproduktivní účinek.



Obrázek 19: Rotační pohyb



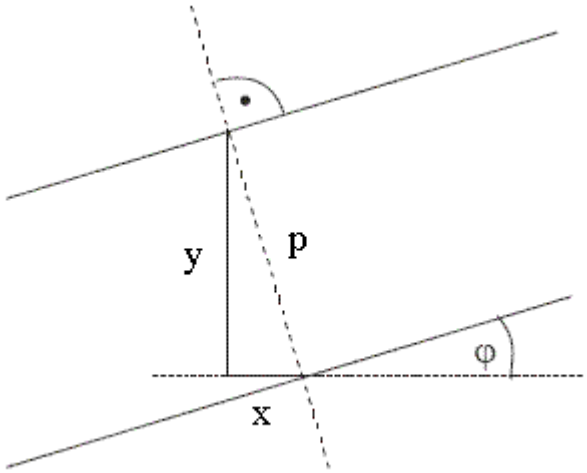
Obrázek 20: Rotační pohyb se vzdálenou osou otáčení

V některých případech může být rozlišení translačního a rotačního pohybu poměrně problematické. Na obrázku Obrázek 20 je znázorněn případ rotačního pohybu s velkým poloměrem otáčení (přibližně půl metru). Díky toleranci při filtrování vektorů podle délky u úhlu je většina vektorů označena za „shodné“, v tomto případě se pohyb označí za translační. Koresponduje to s logikou, že kruh o velkém poloměru lze aproximovat pomocí mnohoúhelníku. Vhledem k tomu, že translační pohyb se zpracovává přesněji (díky filtraci nepovedených odhadů), je v problematických případech označení pohybu za translační ponecháno.

Na tuto nevýhodu je nutné brát zřetel zejména při umisťování kamery na sledovaný objekt. Pokud se sledovaný objekt často otáčí na místě, kamera by měla být umístěna co nejblíže ose otáčení. Pro správný odhad pomalého otáčivého pohybu o velkém poloměru, by bylo vhodné provést vyšší zpracování trajektorie, to znamená zpracovat algoritmus, který je schopen sérii translačních pohybů aproximujících rotační pohyb rozpoznat jako rotační pohyb.

7.8 KALIBRACE

Zatím všechny výpočty probíhaly pouze v obrazové rovině, odhad posunu obrazu v pixelech je již znám. Ale to není hledaný výsledek, to podstatné je o kolik se posunul skutečný objekt. Aby bylo možné provést tento přepočet, je nutné znát převodní koeficient. Ten se zjistí pomocí kalibrace. Pro kalibraci byl vybrán šachovnicový vzor, jeho výhodou je, že je možné, provést kalibraci v ose X i Y z jednoho snímku.



Obrázek 21: Naznačený princip kalibrace ve směru osy Y

Pro kalibraci je použita funkce OpenCV *cvFindChessboardCorner,* která provede vyhledání rohů šachovnice s přesností na jeden pixel. Tato funkce vužívá k lokalizaci rohů Harrisův detektor rohů. Tímto jsou získány souřadnice rohů šachovnice.

Při kalibraci podle osy Y se body seřadí podle hodnoty X. V seřazeném poli se provede výpočet vzdáleností (euklidovovská vzdálenost) mezi jednotlivými body a výpočet úhlů pomocí funkce tangens. Vzhledem k tomu, že jsou body z 2D prostoru transformovány pouze do 1D pole, při přechodu mezi sloupci došlo k určitě k chybě. Případně vlivem zkreslení se může chyba vyskytnout i u pár dalších bodů. Tyto nepřesnosti se odstraní pomocí výpočtu mediánu ve výsleném poli vzdáleností a poli úhlů. Nyní se vypočítá velikost čtverce, v ose Y na obrázku Obrázek 27 označeno jako *y*.

$$y\_px = cos(median\_\varphi) * median\_p \quad (7.3)$$

$$Y\_mm = cos(median\_\varphi) * P\_mm \quad (7.4)$$

Kde y_px je velikost usečky ve směru Y, median_ φ je vypočtený medián úhlu, medián_p je medián velikostí úseček, Y_mm je velikost usečky y v milimetrech a P_mm velikost čtverce kalibračního vzoru v milimetrech.

Výsledný převodní koeficient Koeficient_Y se vypočte jako:

$$Koeficient\_Y = Y\_mm/y\_px \quad (7.5)$$

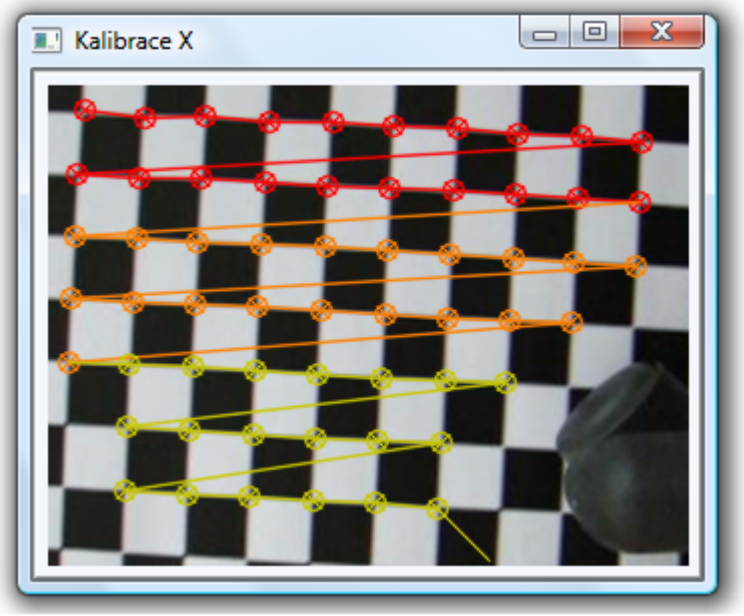
Udává kolik milimetrů odpovídá na jeden pixel v ose Y.

Kalibrace ve směru X probíhá obdobným způsobem.

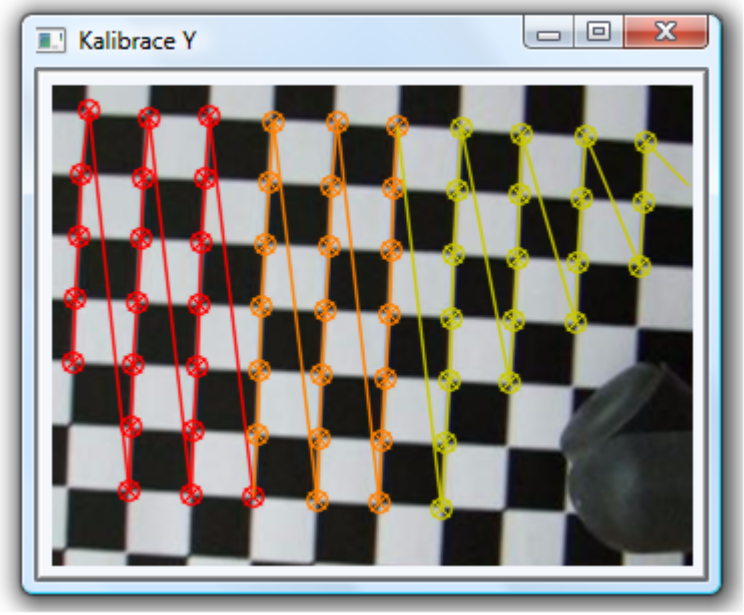
7.9 OBRÁZKY Z KALIBRACE

7.9.1 Správné kalibrační vzory

Následující dva obrázky ukazují vhodné kalibrační snímky (kalibrační výřezy). Úsečky vedoucí po hranách vzoru značí správně provedenou kalibraci.



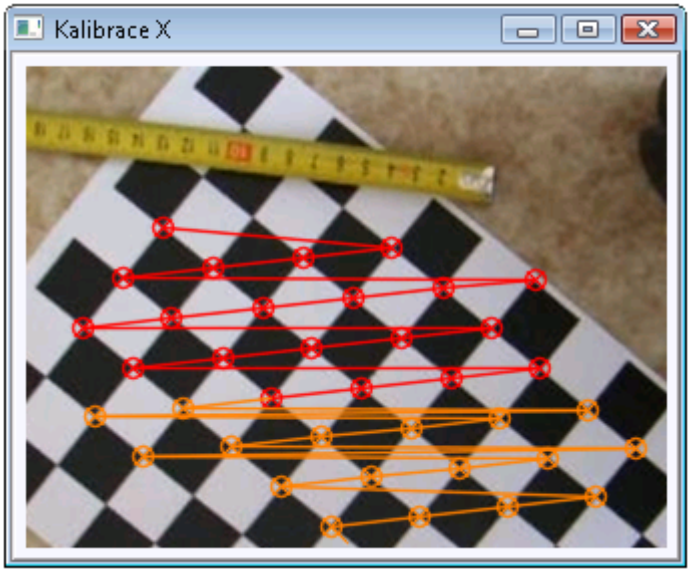
Obrázek 22: Kalibrace ve směru X



Obrázek 23: Kalibrace ve směru Y

7.9.2 Nesprávný kalibrační vzor

Vzhledem k použitému způsobu (řazení bodů podle jedné spuřadnice), není možné provést kalibraci pokud je šachovnice natočena pod úhlem 45°.



Obrázek 24: Kalibrace ve směru X - nesprávná

7.10 ZÁZNAM POHYBU

V tuto chvíli je již vypočten odhad pohybu v osách X a Y. Mezi každými dvěmi vstupními snímky je kromě toho počítán úhel natočení tělesa relativní (popisuje relaci jen mezi posledními snímky) a absolutní úhel natočení (zaznamenává změnu úhlu od výchozího natočení).

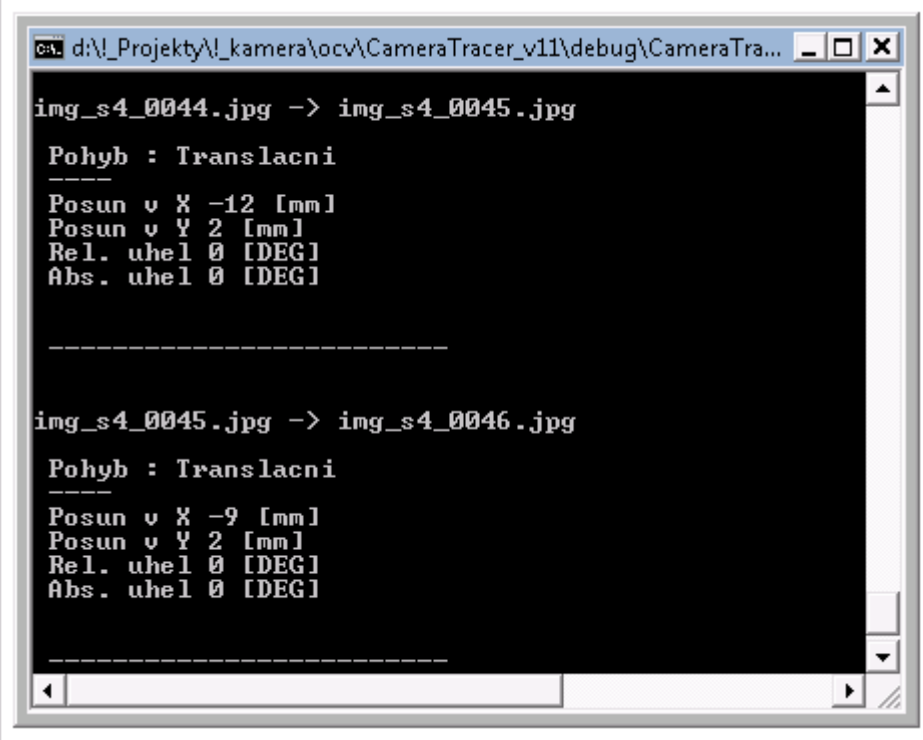
Podle převodních koeficientů se tento odhad přepočítává na skutečný pohyb kamery (objektu).

$$Tx\_mm = Koeficient\_X * Tx \qquad (7.6)$$

$$Ty\_mm = Koeficient\_Y * Ty \qquad (7.7)$$

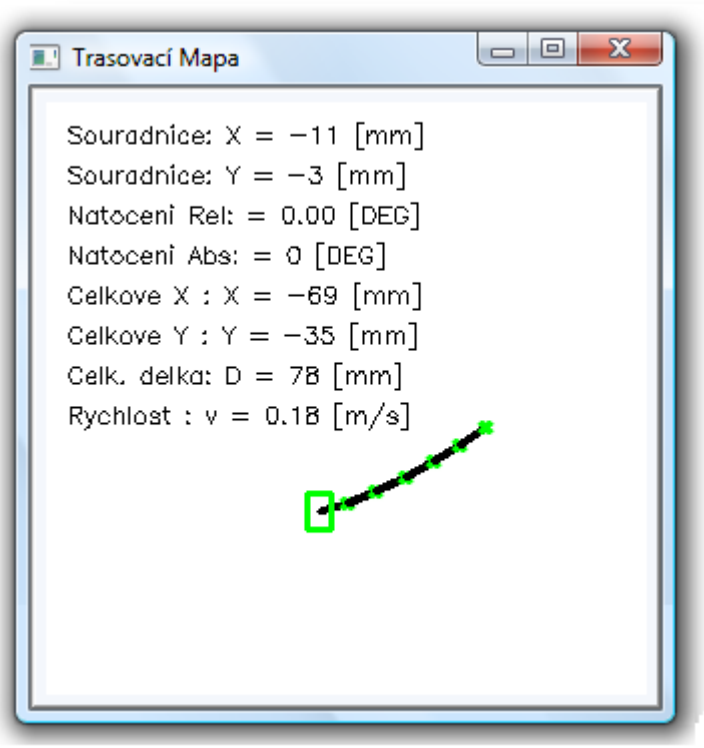
Kde *Koeficient_X* a *Koeficient_Y* jsou kalibrační koeficienty vyjadřující přepočet z pixelů na milimetry. *Tx* a *Ty* jsou zjištěné posuny v pixelech.

Do konzolového okna se vypisuje: jméno prvního souboru a jméno druhého souoboru, zjištěný typ pohybu, posuny v ose X a Y v milimetrech a absolutní a relativní úhel ve stupních (tj. pro snadnější čitelnost, samotný program pracuje s radiány).



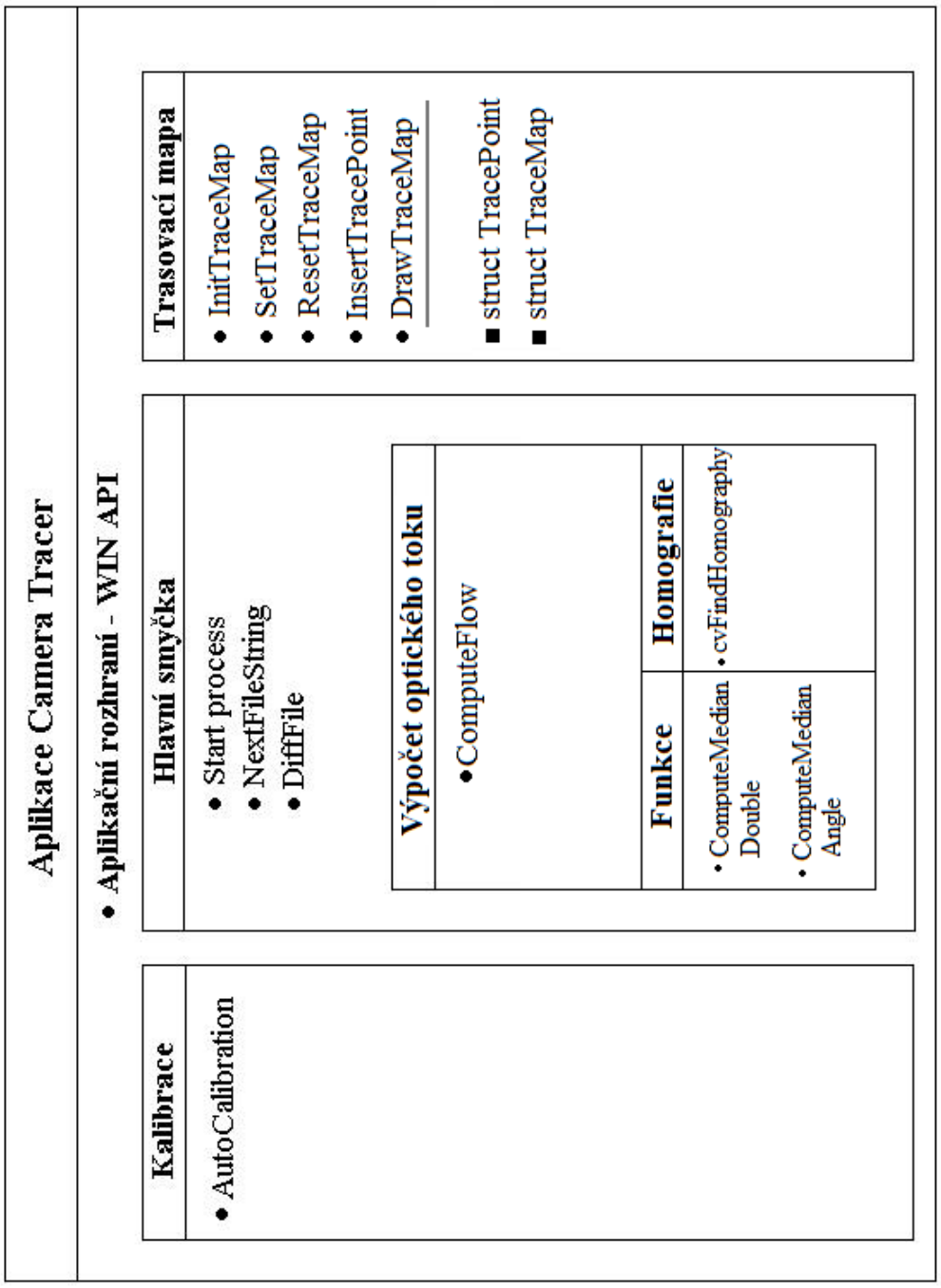
Obrázek 25:Výpis do konzolového okna

Dále se z relací mezi obrázky vytváří trasovací mapa, do ní se ukládá sekvence trasovacích bodů (viz elektronická příloha na DVD - programová dokumentace). Do této sekvence se přidává každý zjištěný pohyb, způsob ukládání je zásobníkový, nový prvek se přidá vždy na začátek a pokud se dosáhne maximální možné délky sekvence je poslední prvek při přidání nového smazán. Struktura je poměrně rozsáhlá a ukládání všech posunů by neúměrně zatěžovalo paměť. Počet ukládaných prvků je nastaven na 150. Tato sekvence se takém v každém kroku vykresluje jako obrázek trasy a výpis parametrů. Vykreslování začíná od prvního prvku (posledního přidaného) a pokračuje dalšími prvky. Kromě posunů a natočení (jako v konzoli) se ještě vypisuje celková trajektorie (pozice) v osách X a Y. Dále celková trajektorie spočtená jako suma všech délek vektorů posunu. Tyto hodnotu platí od počátku, případně od posledního „resetu“ mapy. Z délky posledního posunu a zadaného parametru FPS se spočítá odhad aktuální rychlosti.



Obrázek 26: Trasovací mapa

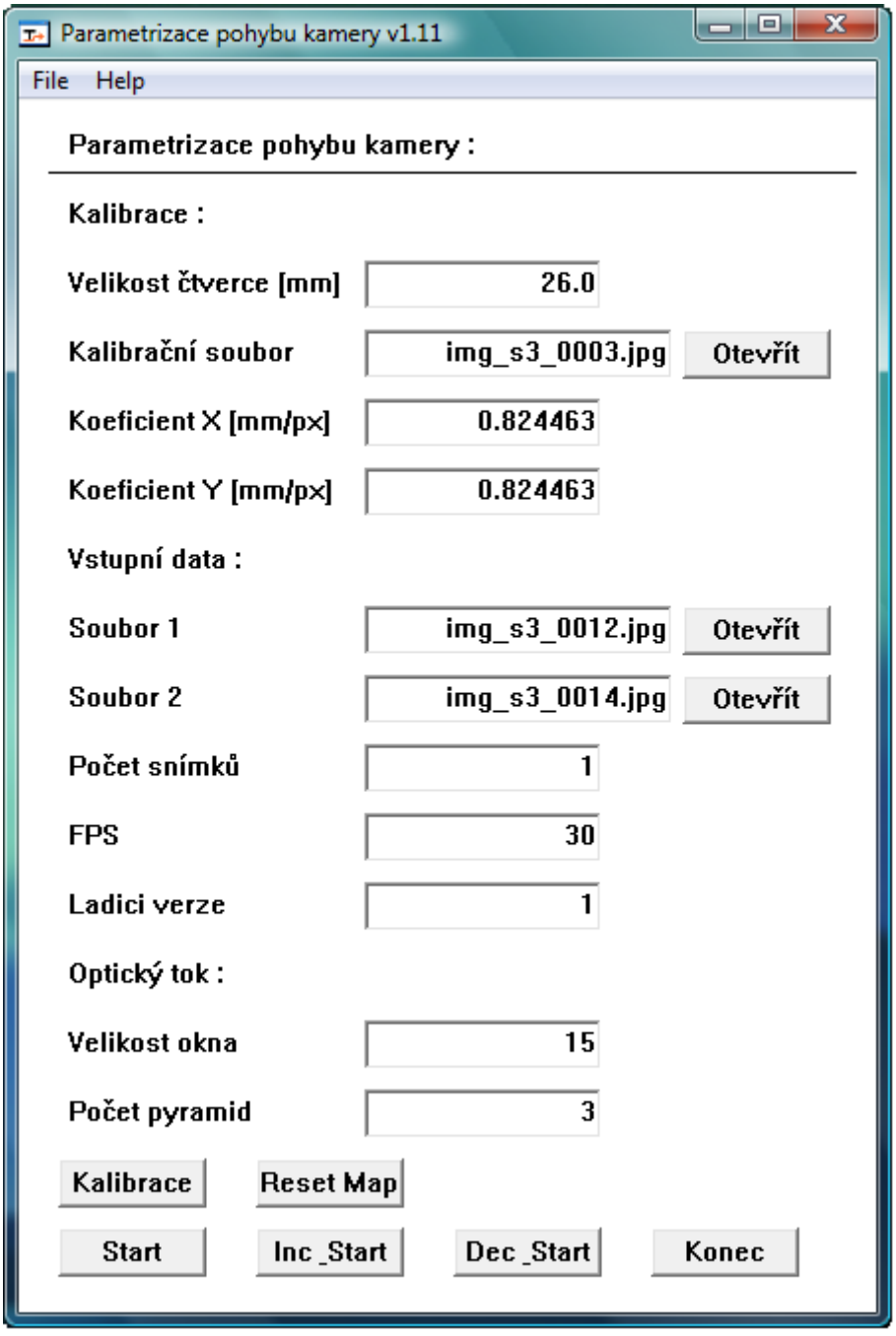
8. POPIS TESTOVACÍ APLIKACE:



Obrázek 27: Struktura programu CameraTracer

Na obrázku Obrázek 27 je přehledové schéma struktury programu CameraTracer. Celé hlavičky funkcí a stručné popisy funkcí jsou uvedeny na DVD - programová dokumentace.

8.1 NASTAVENÍ PROGRAMU:



Obrázek 28: Hlavní okno testovací aplikace

1.sekce: Kalibrace:

Velikost čtverce – velikost čtverce kalibračního vzoru, zadává se v milimetrech

Kalibrační soubor – soubor, pomocí kterého se provádí kalibrace

Koeficient X – přepočtový koeficient, určující kolik milimetrů připadá na jeden pixel v ose X

Koeficient Y – přepočtový koeficient, určující kolik milimetrů připadá na jeden pixel v ose Y

2.sekce: Vstupní data:

Soubor1, Soubor2 – načítané vstupní obrázky, pro jejich výběr slouží tlačítko Otevřít

Počet snímků – určuje kolik snímků se má sekvenčně zpracovat

FPS – Frame Per Second, počet snímků za sekundu, označuje snímkovací frekvenci, s jakou byly snímky pořízeny

Ladící verze – Označuje zda se má program spustit v ladicím režimu (zobrazuje více informací)

3.sekce: Optický tok:

Velikost okna – velikost čtvercového integračního okna, ohraničující okolí bodu pro výpočet optického toku (musí být liché číslo **!**)

Počet pyramid – počet zpracovávaných pyramid algoritmem optického toku (hloubka L)

Tlačítka :

Kalibrace – provede automatickou kalibraci, zjištění koeficientů X a Y pomocí souboru Kalibrační soubor, výsledné hodnoty se zapíší do příslušných polí

Reset Map – provede reset trasovací mapy, vymaže zaznamenanou sekvenci trasovacích bodů, vymaže zobrazenou trasovací mapu, vynuluje počítadla

Otevřít – vybrání snímků pro zpracování pomocí dialogového okna otevření souboru

Start – provede sekvenční zpracování pro zadaný počet snímků

Inc_Start – nejdříve provede inkrementaci indexů obou souborů a poté zpracuje jednu dvojici snímků, zdánlivě opačné pořadí operací je dáno jednodušším laděním, indexy souborů pořád ukazují aktuální hodnoty (nedochází k post inkrementaci)

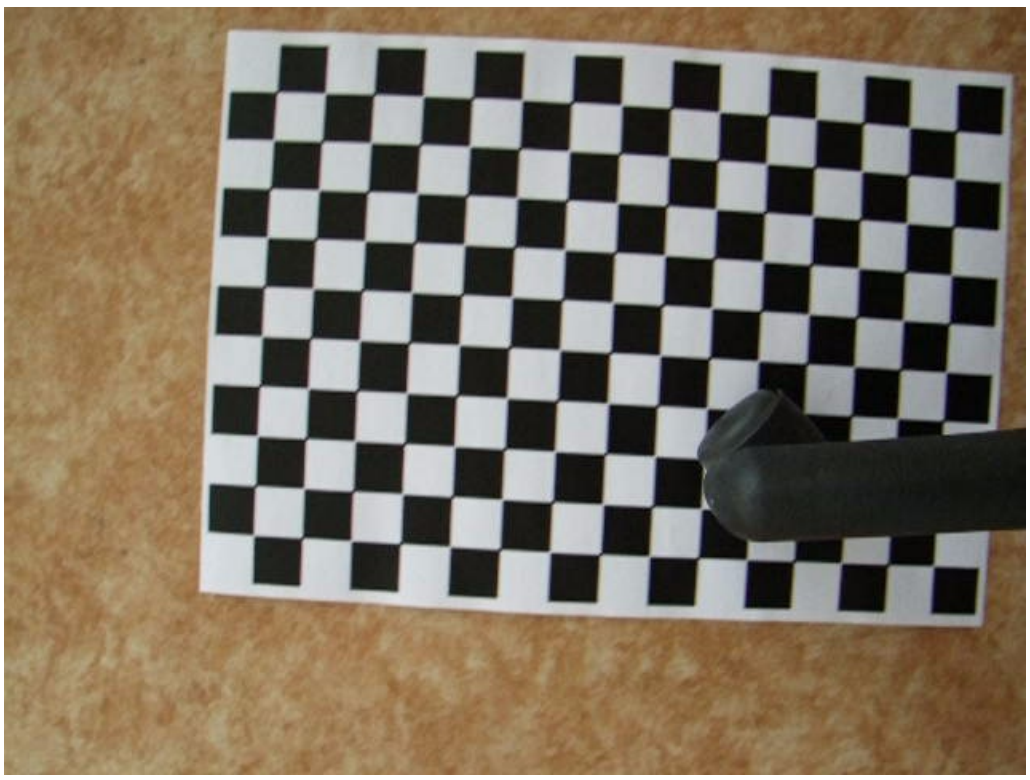
Dec_Start – nejdříve provede dekrementaci indexů obou souborů a poté zpracuje jednu dvojici snímků

Konec – ukončí aplikaci

8.2 OBSLUHA PROGRAMU CAMERATRACER

8.2.1 Kalibrace

První krok, který je nutné udělat na začátku práce je kalibrace, pomocí které se zjistí přepočtové koeficienty z pixelů na milimetry v osách X a Y. Provede se pomocí snímku ze snímané scény, obsahujícího uprostřed šachovnici kalibračního vzoru. Pro kalibraci se používá prostřední polovina obrazu (výřez z prostředku obrazu) na výšku i na šířku. Šachovnice musí mít ve vodorovném i svislém směru stejnou velikost čtverců, samotná velikost čtverců závisí na parametrech objektivu a na snímané výšce. Je doporučeno, aby počet čtverců v prostřední části obrazu byl 50 až 100, příliš velké množství malých čtverců nebo naopak pouze několik velkých čtverců může způsobit nepřesnost kalibrace. Kalibrační šachovnice musí být ve snímku viditelně natočena buď na výšku nebo na šířku, pokud je natočena o úhel blízky 45°, kalibrace se neprovede správně. Na obrázku Obrázek 29 je znázorněn vhodný způsob kalibrace, kalibrační vzor má velikost čtverce 26 mm, scéna je snímána z výšky 60 cm, kalibrace se provádí pomocí výřezů (viz obrázek Obrázek 22: Kalibrace ve směru a obrázek Obrázek 23: Kalibrace ve směru). Zmíněný kalibrační vzor je v příloze.



Obrázek 29: Vstupní obraz do kalibrace

Kalibrace jako taková se nemusí provádět před každým zpracováním, pokud již kalibrace pro danou scénu (nastavení kamery) byla provedena někdy dříve, případně pomocí jiného způsobu (programu), stačí napsat správné převodní koeficienty do formuláře.

8.2.2 Práce s programem

Na začátku práce je nutné zkontrolovat, zda jsou vyplněna políčka koeficient X a koeficient Y, a jestli dávají smysl. Pokud má snímač stejné rozlišení v ose X i Y a ke kalibraci byl použit správný kalibrační vzor, potom by obě čísla měla být přibližně stejně velká. Pokud selhala kalibrace, pokuste se udělat kalibraci znova pomocí jiného snímku.

Vstupem do programu je dvojice snímků ve formátu JPG (ve formuláři označeny soubor 1 a soubor 2), jejich označení se provede pomocí tlačítka otevřít a následného vybrání souborů pomocí dialogového okna otevřít soubor. Pokud se provedlo načtení jména prvního souboru, u druhého souboru stačí napsat jeho jméno (do formuláře) bez použití dialogové nabídky Otevřít soubor. Soubory musí být ve formátu ”prefix”+”NNNN”, kde N je 4-místné číslo (doporučeno je využití programu AviToJpg - viz kapitola 9), musí pocházet ze stejné sekvence a nesmí mezi nimi být příliš velký posun v prostoru. Pomocí políčka počet snímků se zvolí, kolik snímků se má sekvenčně zpracovat, to znamená, že po zpracování první dvojice snímků se u obou snímků inkrementuje číslo souboru a pokračuje se v kontinuálním zpracovávání, dokud se nedosáhne počtu inkrementací daných parametrem počet snímků nebo konce sekvence a další soubory již není možné načíst.

V programu je zakomponována testovací volba, která umožňuje zpracování každého n-tého souboru, pokud například jako první soubor je načten obrázek im_0000.jpg a jako druhý obrázek im_0002.jpg. Před inkrementací indexu souboru se zjistí rozdíl indexů označených souborů, v tomto případě rovno dvěma a zvýšení indexů se provede o hodnotu dvě, provádí se opakovaně pro zvolený počet inkrementací. Pomocí této možnosti lze například simulovat sníženou snímkovací frekvenci z 30FPS na 15FPS. Nastavení většího rozdílu indexů než je hodnota dvě,

případně u pomalého pohybu tři, se ve většině případů nedoporučuje, neboť vede k velkému nárůstu optického toku mezi snímky (velký posun).

Parametr FPS označuje, s jakou snímkovací frekvencí byl pořízen záznam pohybu, pomocí tohoto parametru se vypočítává rychlost pohybu. Pokud snímkovací frekvence není stabilní, vede to pouze k posunutému odhadu rychlosti, na samotný běh programu nemá vliv.

Volba Ladicí verze umožňuje nastavit množství detailů, které program zobrazuje. Program vypisuje výsledky do konzole při libovolném nastavení parametru Ladící verze. Při nastavení čísla nula - nezobrazuje žádné obrázky (výstupem pouze konzole). Nejlepší je ponechat nastavení na čísle jedna - zobrazuje obrázek se scénou a naznačenými optickými toky (jedna výsledná šipka v každém bloku, tj. 25 šipek) a zobrazuje trasovací mapu . Obdobně při nastavení čísla dvě, kde navíc u translačního pohybu vykresluje predikci (zelené úsečky, nakreslené v místech významných bodů). Pokud je zadáno číslo tři - obdobně jako nastavení jedna, ale navíc zobrazuje všechny odhadnuté optické toky (tj. až 250 šipek).

Dále jsou ve formuláři parametry velikost okna (vždy musí být liché číslo!) a počet pyramid, ty je doporučeno nechat nastaveny tak jak jsou, jejich význam je vysvětlen v odstavci Možné optimalizace.

Nyní je možné přistoupit k samotnému spuštění programu, to se provede tlačítkem Start, vhodné je nastavit sekvenční zpracování většího počtu snímků pomocí volby Počet snímků. Jiná možnost je vykonávání výpočtu snímek po snímku, k tomu slouží tlačítka Inc_Start a Dec_Start, ty provedou preinkrementaci případně predekrementaci a poté provedou výpočet pro jednu dvojici snímků. Pro vynulování trasovací mapy slouží tlačítko Reset. Tlačítkem Konec se program zavře.

8.2.3 Možné optimalizace

Parametry velikost okna a počet pyramid, tvoří pokročilé nastavení programu, neboť přímo ovlivňují vlastní výpočet optického toku. Velikost, míní se tím velikost integračního okna, představuje velikost okolí, ve kterém se počítají prostorové derivace $\boldsymbol{I_x}$ a $\boldsymbol{I_y}$ (viz kapitola 4). Velikost okna silně ovlivňuje rychlost výpočtu. Pro malé okno (11-21 pixelů) je výpočet relativně rychlý, se zvětšujícím se oknem rychlost výpočtu klesá, zároveň se ve většině se zvýší přesnost odhadu (sníží se počet přiřazení k špatným bodům - nekorespondence). Optimální nastavení je mezi 15-25,

záleží na rychlosti pohybu a „rozlišitelnosti“ sledovaného povrchu. Pro povrch s velkým množstvím výrazných bodů je možné volit menší čísla, pro nevýrazné povrchy je lepší zvolit větší číslo, ale ani to obecně nezaručuje správný výsledek. Opět upozorňuji, že parametrem mohou být pouze lichá čísla.

Parametr počet pyramid (viz kapitola 4.2), určuje kolik pyramid se vytváří ze vstupních snímků. Pomocí pyramid je možné rozšířit maximální okolí bodu, ve kterém je možné vypočítat správný optický tok. Parametr má vliv na rychlost výpočtu. Pro běžné výpočty se doporučuje hodnota tři, pro pomalejší pohyb (pokud to bude algoritmus stíhat) je možné zvolit hodnotu dvě. Nastavení na hodnotu jedna způsobí, že se pyramidová implementace neprojeví.

8.3 TESTOVÁNÍ PROGRAMU

Testování aplikace CameraTracer bylo prováděno pomocí snímků s rozlišením 640px na 480px, které byly uloženy z videa natočeného kompaktním fotoaparátem Sony W-5. Později byl použit fotoaparát Fujifilm FinePix S100FS, který má podstatné výhody oproti Sony W-5, protože má kratší ohniskovou vzdálenost 28 mm místo 35 mm a je schopný rychlejšího doostřování. Zorný úhel fotoaparátu Fujifilm je při nejkratší ohniskové vzdálenosti 28 mm rovný 75,3°. Nevýhodou je, že ani tento fotoaparát neumožňuje pevné zaostření na určitou vzdálenost. V určitých případech má ostřící automatika velký problém správně zostřit, například leštěné kamenné povrchy s drobným vzorem ji lehce zmatou.

Při natáčení testovacích videí byla použita obyčejná kancelářská židle a na ní byl připevněn stativ s fotoaparátem (obrázek Obrázek 30) ve výšce 60 cm nad zemí. Výhodou vyššího umístnění snímače je snížení rychlosti optického toku, to umožňuje dosáhnout správného zpracování i při větší rychlosti pohybu sledovaného objektu. Téměř všechna videa byla natáčená při nastavené ohniskové vzdálenosti na 28mm z výšky 60cm.

Testovací sekvence jsou k dispozici na přiloženém DVD. V každé složce jsou sekvence obrázků získané z natočených videí, dále původní videosoubor sloužící pro náhled. Dále obsahuje textový soubor stručně popisují způsob získání záznamu, obsahuje vhodné nastavení aplikace a upozorňuje na zajímavé nebo problematické pasáže.

Obrázek 30: Záběr na fotoaparát při snímání scény

8.4 OVĚŘENÍ PŘESNOSTI ZAZNAMENANÉ TRAJEKTORIE

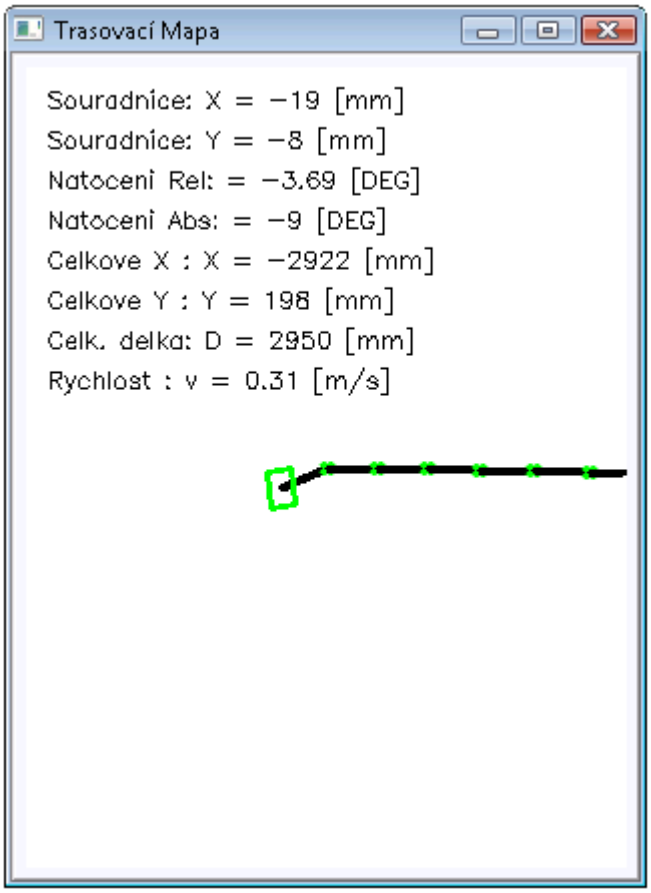
Záznam byl natočen naznačeným způsobem (viz předchozí kapitola), podrobněji rozepsáno v textovém souboru na DVD. Fotka snímané scény je na předchozím obrázku.

Nastavení programu při testu bylo následující :

Koeficient X –	0.825
Koeficient Y –	0.825
Soubor1 –	img1_0085.jpg
Soubor2 –	img1_0087.jpg
Počet snímků –	152
FPS –	15
Ladící verze –	1 (nemá vliv na vlastní výpočet)
Velikost okna –	21
Počet pyramid –	3

V sekvenci je zaznamenám převážně translační pohyb, židle na které byla umístěna kamera se pohybovala po linoleu podél nataženého tří metru. Na prvním snímku (img1_0085.jpg) je na jeho pravém okraji přibližně začátek tří metru. Druhý snímek (img1_0087.jpg) má oproti prvnímu snímku rozdíl indexů roven dvě, to znamená že se bude inkrementovat vždy o index dvě - tj. zpracují se pouze liché soubory. V tomto případě je to možné, zaznamenaný pohyb je relativně pomalý. Počet snímků je nastaven na 152 - po tolika inkrementacích se dosáhne konce tří metru. FPS je nastaveno na 15, video bylo natočeno se snímkovací frekvencí 30, ale zpracovává se pouze polovina snímků. Velikost okna je nastavena na 21, protože linoleum nemá moc výrazný vzor. Počet pyramid je nastaven na hodnotu tři. Tato testovací sekvence je rovněž obsažena na přiloženém DVD v adresáři Testovací sekvence ve složce img_s01.

Po proběhnutí popsaného postupu, se zobrazí následující Trasovací mapa:

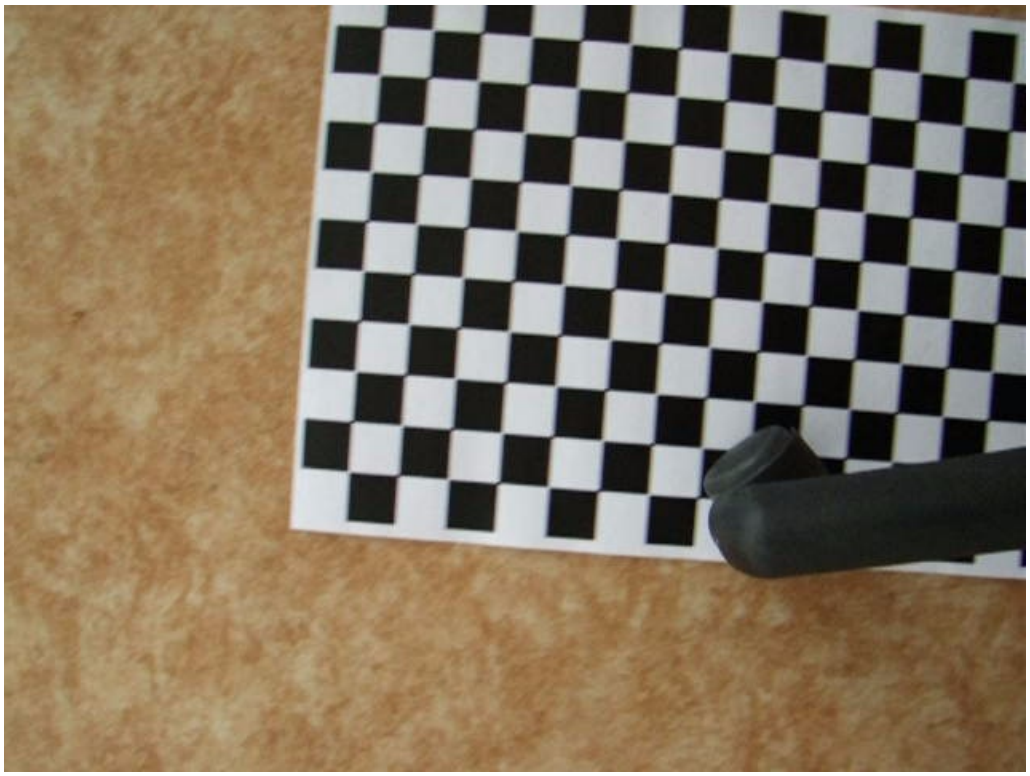


Obrázek 31: Trasovací mapa - test

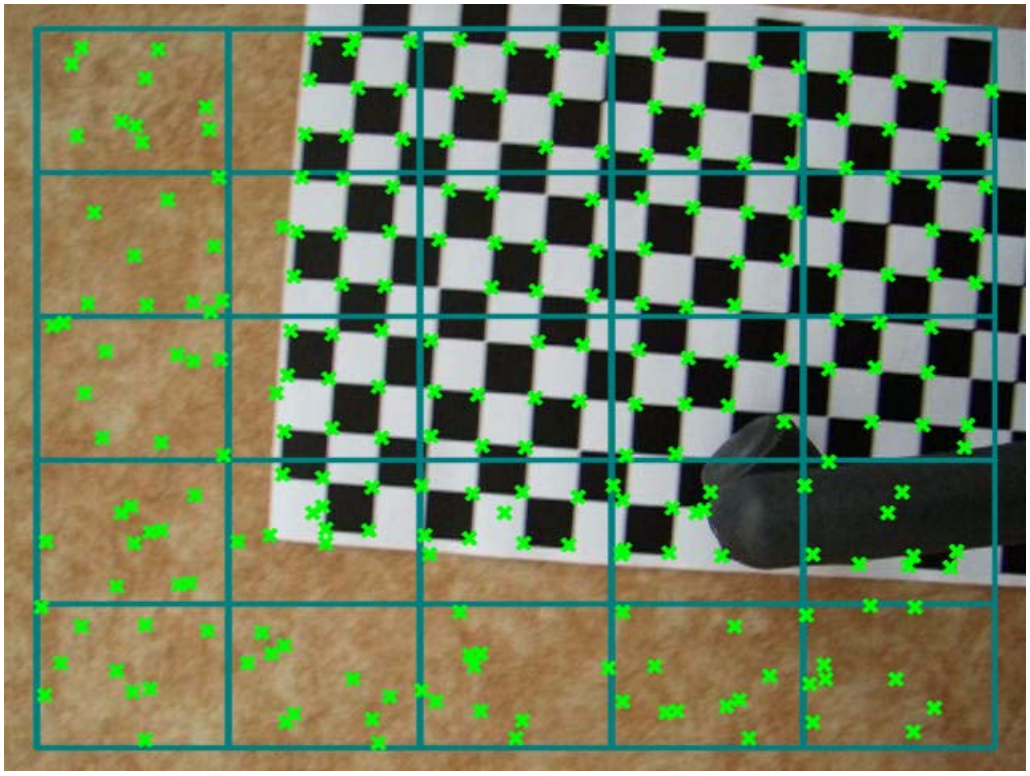
Předchozí obrázek ukazuje, že změna v ose X je -2922 mm (znaménko mínus je proto, že pohyb doleva byl označen za záporný), z toho vyplývá odchylka o 78 mm naměřená na třech metrech. Celková naměřená trajektorie je 2950 mm, to ukazuje, že pohyb ve směru osy Y byl téměř zanedbatelný. Vzhledem k tomu, že při testu byl fotoaparát umístěn na stativu (který byl upevněn na kancelářské židli) bez přesné aretace (pouze na základě optické kontroly) je získaný výsledek velmi přesný.

8.5 OBRÁZKY Z TESTOVACÍ APLIKACE

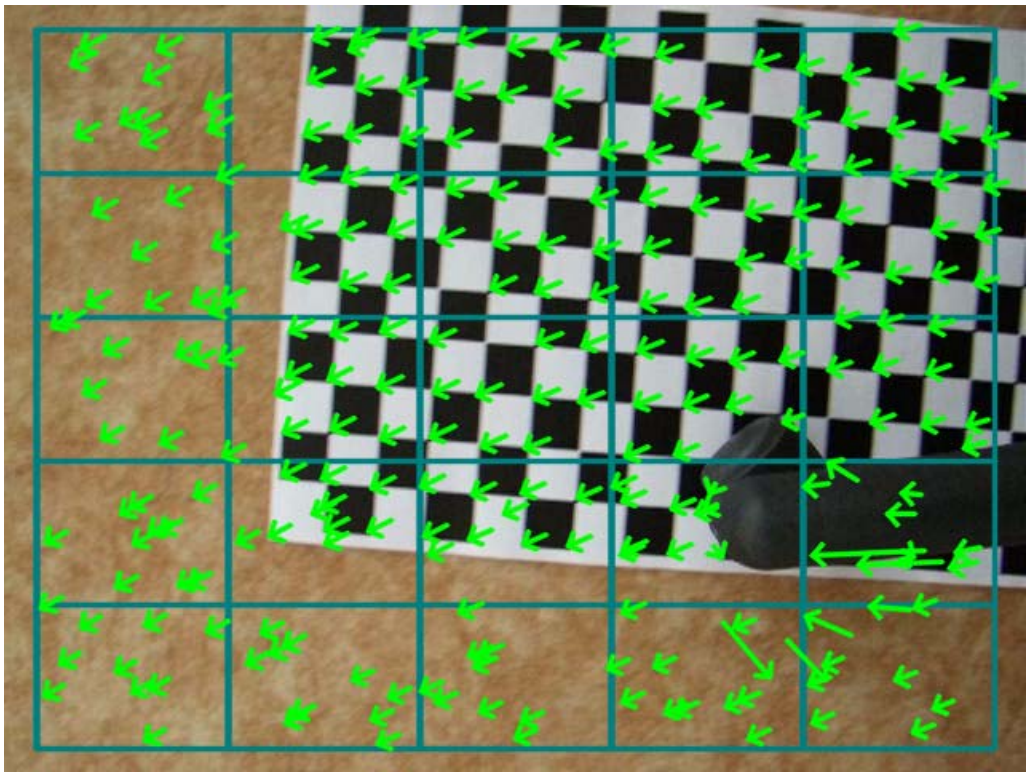
Obrázek 32: Vstupní soubor 1



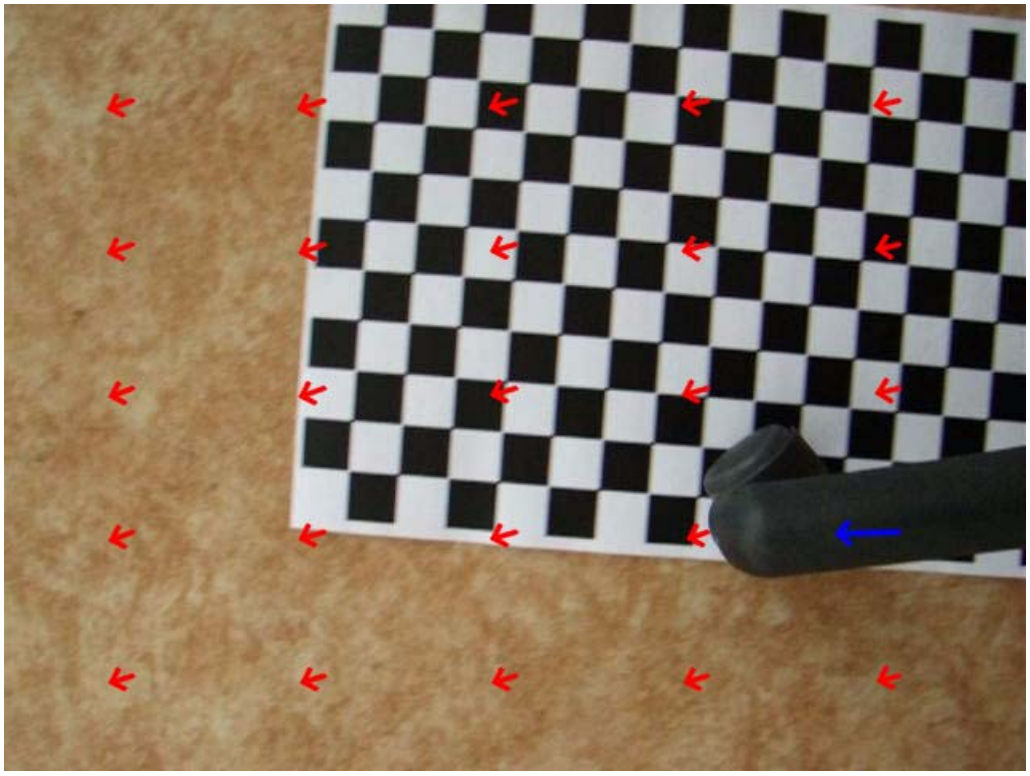
Obrázek 33: Vstupní soubor 2



Obrázek 34: Nalezené významné body v jednotlivých polích



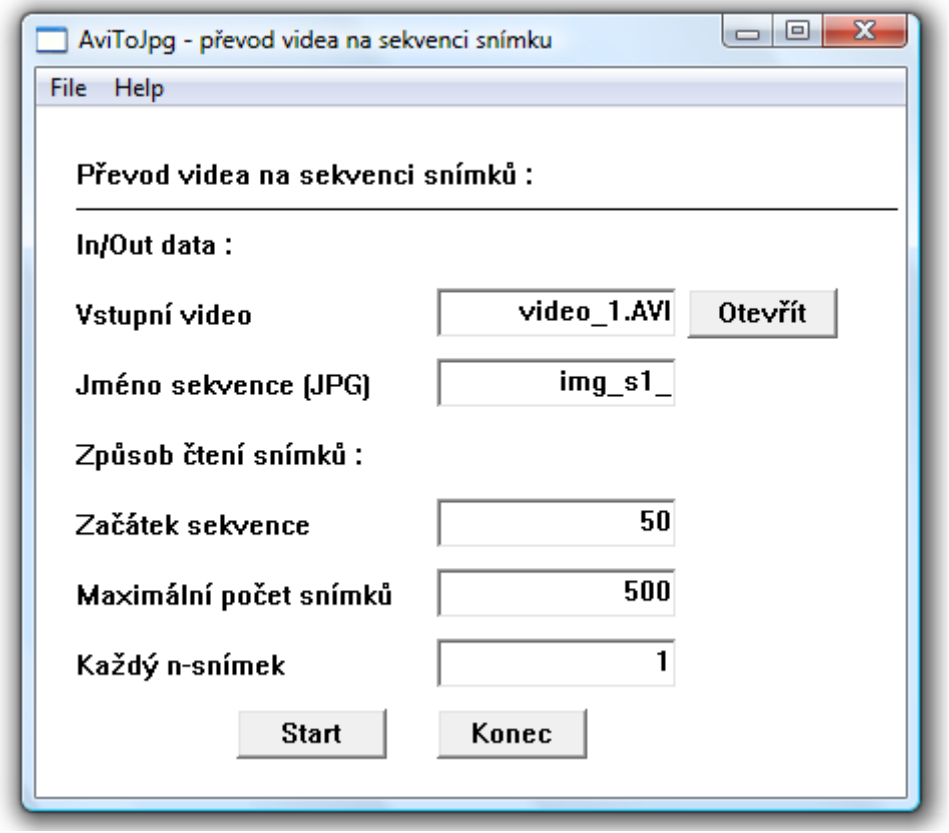
Obrázek 35: Optický tok



Obrázek 36: Zjištěné optické toky v jednotlivých blocích

9. POMOCNÝ PROGRAM AVI TO JPG

Program CameraTracer pracuje offline tak, že postupně načítá jednotlivé dvojice obrázku ve formátu JPG. Nejprve je nutné natočit video ve formátu AVI a poté ho převést na sekvenci snímků. Právě k tomu slouží program AviToJpg. Ten je vytvořen podobně jako program CameraTracer v C++ využívajíc knihovny OpenCV. Rozhraní programu je ve WIN API.



Obrázek 37: Aplikace AviToJpg

9.1 POPIS APLIKACE AVI TO JPG

Vstupní video – vstupní soubor ve formátu AVI

Jméno sekvence (JPG) – prefix sekvence obrázků ve formátu JPG,
výstupní formát : ”prefix”+”NNNN”, kde N je 4-místné číslo

Začátek sekvence – číslo snímku ve vstupním videu, od kterého se začnou sekvenčně ukládat obrázky

Maximální počet snímků – určuje kolik snímků se má ze vstupního videa uložit, pokud je sekvence kratší, skončí se s posledním načteným snímkem

Každý n-snímek – určuje, s jakou frekvencí se mají snímky ukládat, číslo jedna znamená každý snímek, číslo dvě znamená každý druhý snímek

Převod se spustí stisknutím tlačítka Start, poté se otevře nové okno, ve kterém se bude zobrazovat průběh konverze, respektive aktuální ukládaný snímek, tak jak se prochází video souborem.

Práce s programem je velmi intuitivní. Jediné čemu je nutné přikládat větší pozornost je vstupní soubor, který musí být ve formátu AVI a v podporovaném kodeku. Pro načtení videa je využita funkce OpenCV. Ta podporuje následující kodeky: Indeo Video 5.10, Intel Indeo Video R3.2, Cinepak Codec by Cti a kodeky typu MJPG (motion-jpeg) a samozřejmě lze použít i bezztrátové AVI. Pokud vstupní video není kódováno vhodným kodekem, je nutné provést převod do vhodného formátu externím programem, například pomocí VirtualDub.

Pokud se dodrží stejný formát výstupního jména (”prefix”+”NNNN”, kde N je 4-místné číslo) souboru, je možné použít pro převod videa na sekvenci obrazů typu JPG libovolný software.

10. POZNATKY Z TESTOVÁNÍ

Program nejprve provádí vyhledání významných bodů, k tomu je použit algoritmus GoodFeatureToTrack, který jsem upravil tak, aby došlo k většímu rozprostření bodu ve snímku rozdělením na více oblastí (viz 7.3.1). Tato úprava se ukázala jako velmi výhodná, jednak se tím najde větší počet různých významných bodů, ale hlavně nedochází ke shlukování významných bodů v jedné části snímku. V průběhu testování vyšlo jako vhodné řešení rozdělení na 25 oblastí. Navíc tento princip umožňuje při translačním pohybu následné vyfiltrování oblasti, pokud se v ní vyskytne pohyb odlišný od celkového pohybu.

Optický tok pro označené významné body se počítá metodou Lukas-Kanade. Ta je velmi spolehlivá pro velké množství snímků, samozřejmě (jak vyplývá z kapitoly 4.1), je důležité, aby světlost scény zůstávala co nejvíce konstantní, případně se měnila jen velmi zvolna, ne skokově, protože optický tok je založen na lokálních jasových vlastnostech. Dále je přesnost výpočtu optického toku větší pro menší pohyby a rovněž rychleji konverguje, to je dáno iterativním výpočtem Newtonovou metodou. Výhodnější je širší záběr scény, v záběru bude méně detailů, ale objekty ve scéně budou mít menší velikost optického toku.

Robustnost nalezení správného posunu má zajistit počítání mediánu z nalezených vektorů optického toku v každé oblasti, tak vznikne v každé oblasti jeden vektor. Tím, že se u translačního pohybu aplikuje mediánová filtrace a vypouští se chybné (outliers) vektory (viz 7.5 Zpracování získaných dat), došlo k velkému zpřesnění odhadu trajektorie. Z vyfiltrovaných vektorů počítá matice homogenní transformace, z té se odečítají podle typu pohybu translační nebo rotační koeficienty.

Výsledkem zpracování jsou získané souřadnice pohybu a trajektorie získaného pohybu zakreslená do trasovací mapy.

Maximální rychlost objektu s jakou je schopný program pracovat je přibližně 0,5 m/s, pokud se objekt pohybuje translačním pohybem bez výraznějších výkyvů ve směru pohybu, maximální akceptovatelná rychlost stoupne až na dvojnásobek, to znamená 1 m/s (díky predikci pohybu). Přesnost určení pohybu při převažujícím translační pohybu je přibližně +/- 3 cm na jeden metr posunu. Pro rotační pohyb je

přesnost horší - přibližně +/- 10° při otočení o 360°. Jde o typické hodnoty, velmi záleží na snímaném podkladu a rychlosti pohybu. Tyto hodnoty byly naměřeny pro povrch z linolea (obrázek Obrázek 30) a povrch venkovních dlaždic (obrázek Obrázek 17), pro rozmanitější povrch by pravděpodobně bylo dosaženo lepších výsledků. Naopak pro některé povrchy je trasování tímto způsobem prakticky nemožné, jako je například na obrázku Obrázek 38, kde je téměř jednolitá barva světlého linolea. A Ani tmavší část linolea na levém okraji obrázku už to nezachrání, výsledkem budou zmatené odhady optického toku a nesmyslná trajektorie pohybu.

Doba zpracování jednoho snímku se pohybuje zhruba kolem 70-120 ms, záleží na nastavení programu, složitosti scény, rychlosti pohybu sledovaného objektu a výkonu počítače. Například pro scénu jako je na obrázcích Obrázek 17 až Obrázek 19 (jejich velikost je 640 px na 480 px), při rychlosti pohybu přibližně 0,3 m/s, na notebooku s procesorem IntelCore2Duo® 2,4GHz, pro sekvenci 300 snímků a pro velikost integračního okna 15, trvá standardní běh programu typicky 20,8 sekundy. Z čehož vyplývá průměrná doba zpracování jednoho snímku 69ms. Pro srovnání pro velikost integračního okna 21 je to hodnota 24,6 s celkově a 82 ms na snímek. Při velikosti integračního okna 25 je to 34,2 s celkově a 114 ms na snímek. Zvolit optimální velikost integračního okna pro libovolný povrch a rychlost posunu je obtížné. Ze série testování vychází nejlépe velikost 15, je to kompromis mezi přesností odhadu pro různé povrchy a rychlostí výpočtu. Většinou platí, že pro méně členitý povrch je lepší větší velikost integračního okna.

Obrázek 38: Nevhodný snímek k trasování

11. ZÁVĚR

Začátek práce je věnován úvodu do problematiky trasování pohybu kamery, zejména se věnuje možnostem jak nalézt vektor posunu mezi dvěma snímky. Další část je věnována hledání význačných bodů v obraze s ohledem na navazující popis algoritmu GoodFeaturesToTrack a jeho využití. Cílem vyhledávání významných bodů v obraze je zmenšit objem zpracovávaných dat a připravit data pro algoritmus „*sparse optical flow"*, konkrétně metodu Lukas-Kanade, pomocí které je počítán optický tok. Tato část programu je velmi robustní, pracuje s velkým množstvím různých scén. Od scén, kde je možné najít velké množství výrazných bodů až po scény, které působí hodně kompaktně (například deska stolu či venkovní beton), u nich je velmi obtížné sledovat pohyb i lidským okem.

Další část práce se zabývá zpracováním získaných odhadů optických toků, od filtrace dat pomocí mediánu, který má odhalit „outliers" u translačního pohybu, přes výpočet transformací mezi snímky pomocí matice homogenní transformace, která zachycuje změny mezi snímky v afinním prostoru. Výsledkem zpracování jsou souřadnice X a Y popisující změny mezi snímky a trajektorie pohybu.

Správná funkčnost popsaného algoritmu byla ověřena pomocí programu CameraTracer, který je naprogramován v C++ a využívá knihovnu OpenCV. Program byl testován pomocí snímání různých scén a pohybů, byly zjišťovány výhody a naopak nevýhody a omezení daného postupu. Podrobnějšímu popisu výsledků získaných z testování se věnuje předchozí kapitola.

12. SEZNAM POUŽITÉ LITERATURY:

[1] Šonka M., Hlaváč V.: Počítačové vidění GRADA, Praha 1992, ISBN 80-85424-67-3

[2] Haußecker H., Geißler P.: Handbook of Computer Vision and Applications. San Diego: Academic press, 1999, Volume 2, str. 309-417.

[3] Jianbo Shi and Carlo Tomasi, „Good features to track“, Proc. IEEE Comput. Soc. Conf Comput. Vision and Pattern Recogn., pages 593-300,1994.

[4] J.-Y. Bouguet, "Pyramidal implementation of the lucas kanade feature tracker," Intel Corporation, Microprocessor Research Labs, Tech. Rep., 2000.

[5] OpenCV - Introduction [online], poslední revize 5.4.2006. [cit. 14.1.2007] Dostupné z < http://www.intel.com/technology/computing/opencv/ >

[6] OpenCV Wiki [online], poslední revize 2009-03-18 [cit. 7.12.2008]. Dostupné z < http://opencvlibrary.sourceforge.net />

[7] http://www.intel.com/software/products/ipp/index.htm - (online)¨

[8] URBAN, M., Harris Interest Operator [online], poslední revize 28.1.2003 [cit. 7.12.2008]. Dostupné z <http://cmp.felk.cvut.cz/cmp/courses/dzo/resources/lecture_harris_urban.pdf>

[9] Learning OpenCV, Dr. Gary Rost Bradski, Adrian Kaehler
ISBN: 9780596516130, 575 stran

[10] Computer Vision: A Modern Approach, David A. Forsyth, Jean Ponce,
ISBN-13: 9780130851987, Vydavatel: Prentice Hall, 2003, 693 stran

[11] Tomáš Svodoba, ČVUT FEL, Centrum strojového vnímání, přednášky, http://cmp.felk.cvut.cz

[12] HLAVÁČ, V., Studijní materiály [online], poslední revize 1.5.2005 [cit. 7.12.2008].
Dostupné z <http://cmp.felk.cvut.cz/~hlavac/Public/Pu/Teaching/>

13. SEZNAM ZKRATEK A SYMBOLŮ

AVI - Audio Video Interleave - multimediální kontejner

FPS - Frame Per Second, označuje snímkovací frekvenci

JPG (JPEG) - Joint Photographic Expert Group, definuje způsob komprese obrazových dat, většinou je tím míněn obrázek uložený tímto způsobem

Lucas-Kanade - v této diplomové práci je takto označován algoritmus optického toku s rozptýlenými body využívající iterační pyramidový algoritmus Lucas-Kanade

OpenCV - Knihovna Open Source Computer Vision Library od Intelu®

14. PŘÍLOHY

Seznam Příloh

Příloha A : Výčet hlaviček použitých funkcí (popis funkcí je uveden v programové dokumentace, která je umístěna na přiloženém DVD)

Příloha B : Obsah DVD

Příloha C : Kalibrační vzor, velikost vzoru 26 mm, stránka A3 (vyjímatelně vloženo)

B. Obsah CD

- Testovací aplikace CameraTracer.exe
- Kompletní zdrojové kódy aplikace CameraTracer
- Programová dokumentace- složka Dokumentace
- Návod na použití aplikace: readme.rtf
- Sada testovacích sekvencí obrázků
- Kalibrační vzor (šachovnice) - složka Kalibrace
    - Soubory: sachovnice_A3.tif a sachovnice.tif
- Instalační prostředky:
    - Open CV (pro spuštění programu není instalace nutná)
- Diplomová práce – elektronická verze